Хмельницький національний університет
Факультет інформаційних технологій
Кафедра автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Автоматизована системи зарядки дронів

КвРАКІТ. 2022131.01.16.ПЗ

Рівень вищої освіти перший

Галузь знань 15 «Автоматизація та приладобудування»
Шифр, назва

Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»
Шифр, назва

Освітня програма «Автоматизація та комп'ютерно-інтегровані технології»
Назва

Виконав:

студент III курсу, група АКІТс-22-1 _____ Владислав РЕМАРЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник _____ Ірина ФОРКУН
Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер _____ Людмила КОРЕЦЬКА
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
зав. кафедри АКІТтаР _____ Валерій МАРТИНЮК
Підпис Ім'я, ПРІЗВИЩЕ

« 16 » червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *автоматизації, комп'ютерно-інтегрованих технологій та робототехніки*
Рівень вищої освіти *перший (бакалаврський)*
Галузь знань *15 – Автоматизація та приладобудування*
Спеціальність *151 – Автоматизація та комп'ютерно-інтегровані технології*
Освітня програма *Автоматизація та комп'ютерно-інтегровані технології*

ЗАТВЕРДЖУЮ
Завідувач кафедри АКІТтаР
Валерій МАРТИНЮК
07. лютого 2025

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Ремарчуку Владиславу Олеговичу
Прізвище, ім'я, по батькові студента

1 Тема роботи Автоматизована системи зарядки дронів.

Керівник роботи Форкун Ірина Валеріївна, к.т.н., доцент
Прізвище, ім'я, по батькові, науковий ступінь, учене звання

Затверджено наказом ректора університету від 07.02.2025р. №23

2 Строк подання студентом роботи на кафедру 02.06.2025р.

3 Вихідні дані до роботи метою виконання роботи є розробка та впровадження системи зарядки акумуляторних дронів для зменшення часу заміни батареї та не активності дрону

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. Аналіз відомих засобів та рішень. Огляд існуючих систем автоматизації процесу зарядки дронів. Проектування автоматизованої системи зарядки дронів. Програмно-апаратна реалізація автоматизованої системи зарядки дронів

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
презентаційні матеріали (слайди)

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Антиплагіат	Федула М.В., доцент кафедри АКІТтаР		
Нормоконтроль	Корецька Л.О., доцент кафедри АКІТтаР		

7 Дата видачі завдання 07 лютого 2025

КАЛЕНДАРНИЙ ПЛАН

Назва розділу кваліфікаційної роботи	Строк виконання	Примітка
Вступ	15.03.2025	виконано
Огляд відомих засобів та рішень	24.03.2025	виконано
Проектування автоматизованої системи зарядки дронів	15.04.2025	виконано
Програмно-апаратна реалізація автоматизованої системи зарядки дронів	18.05.2025	виконано
Висновки	22.05.2025	виконано
Оформлення пояснювальної записки КРБ	01.06.2025	виконано
Оформлення презентаційних слайдів	10.06.2025	виконано

Студент _____ Ремарчук В.О
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Форкун І. В.
Підпис Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Автоматизована системи зарядки дронів».

Автор роботи: Ремарчук В.О.

Керівник роботи: Форкун І. В.

Пояснювальна записка: 57 с., 34 рис., 2 табл., 1 дод., 41 джерел.

Графічна частина: 11 презентаційних слайдів.

ФУНКЦІОНАЛЬНА БЕЗПЕКА, НАДІЙНІСТЬ, РОБОТ МАНІПУЛЯТОР, ПОЗИЦІОНУВАННЯ, АВТОМАТИЗОВАНА СИСТЕМА.

Метою роботи є розробка та впровадження системи зарядки акумуляторних дронів для зменшення часу заміни батареї та не активності дрону. Для виконання поставленої мети було розглянуто наступні завдання. Проаналізовано та обрано способи для зарядки акумуляторних батарей дрону. Проведено аналіз доцільності подібної автоматизації. Розроблено блок-схему керування системою автоматизованої зарядки дрону. Розроблено конструкцію модуля автоматичної заміни акумулятора. Розроблено програмне забезпечення для керування процесом заміни;

Підпис студента

16.06.2025

Дата

ЗМІСТ

ВСТУП

Дрон - безпілотний літальний або наземний апарат, здатний пересуватися самостійно за заданим алгоритмом, без управління людиною або при керуванні дистанційно [1]. Для більшості світу дрони мають не лише військове призначення а й застосовуються в багатьох галузях життя. Як приклад Велика Британія використовує дрони для доставки пошти, Австралія використовує патрульні дрони з вбудованими рятувальними кругами в берегових районах для пошуку людей які тонуть або кораблетрощах, навіть в Україні останніми роками активно використовують дрони для пошуку правопорушників які палять сміття або потенційні місця початку лісових пожеж [2,3]. Найбільше та найкраще дрони себе проявили у аграрній галузі, дрони використовують для обприскування полів замість малогабаритних літаків та спеціальної техніки а також для аналізу росту висаджених рослин та пошуку людей які займаються крадіжками на полях [4,5,6].

Не зважаючи на галузь використання більшість дронів використовують акумуляторні батареї, через що час їх роботи не перевищує години для важких промислових дронів, для простіших дронів час роботи зазвичай складає від п’яти до тридцяти хвилин, що є достатнім для невеликих завдань, але замалим для повноцінного використання дронів для автоматизації та спрощення роботи людини. Як результат користувачі носять з собою запасні акумулятори та кожен раз при необхідності заміни повертають його до себе та вручну виконують заміну. Хоча цей процес неможливо автоматизувати для автоматичної заміни акумуляторних батарей певного типу та зарядки використаного акумулятору без втручання людини. У випадку використання подібної системи у віддаленій місцевості без втручання людини можливо налагодити цілодобовий моніторинг території чи виконання певної дії.

Метою роботи є розробка та впровадження системи зарядки акумуляторних дронів для зменшення часу заміни батареї та не активності дрону. Для виконання поставленої мети слід виконати наступні завдання:

- проаналізувати та обрати способи для зарядки акумуляторних батарей дрону;
- провести розрахунки доцільності подібної автоматизації;
- розробити блок-схему керування системою автоматизованої зарядки дрону;
- розробити конструкцію модуля автоматичної заміни акумулятора;
- розробити програмне забезпечення для керування процесом заміни;
- інтегрувати станцію заміни у безпілотну систему.

1 ОГЛЯД ВІДОМИХ ЗАСОБІВ ТА РІШЕНЬ

1.1 Огляд існуючих систем автоматизації процесу зарядки дронів

Однією з основних технічних проблем, яка впливає на ефективність використання дронів, є обмеження які виникають через використання невеликих акумуляторних батарей вага та об’єм яких впливає на час використання, нажаль використання більших акумуляторів не часто підходить, через вплив більшої ваги акумулятора на вашу дрону, як результат це додаткове навантаження на мотори, які для компенсації збільшення ваги споживають більше енергії що швидше використовує запас енергії та вимагає більш потужних моторів. Як результат при розгляданні доступних варіантів слід враховувати додаткове навантаження від можливих модифікацій та шукати найкращий варіант для внесення мінімальної кількості модифікацій в структуру дрону [7].

Розглянемо основні підходи та реалізації таких систем.

Найпростішим методом автоматизованої зарядки є використання спеціальних зарядних станцій-платформ та невеликих модифікацій в модуль контролю акумулятору. Такі системи використовують контактні елементи або шини на ніжках дрону які контактують з зарядними елементами на посадковій платформі. Як приклад, компанія SkyCharge розробила модифікацію яка кріпиться до ніжок дрону та до акумуляторної батареї (рисунок 1.1) [8]. Дрон приземляється на платформу створюючи контакт після чого система керування перемикає реле та подає живлення на платформі, тим самим виконуючи швидку зарядку від мережі чи джерела.

Перевагами подібних систем є висока ефективність заряджання, простота реалізації, надійність. До недоліків можна віднести високу точність посадки якщо не інтегрувати системи наведення, неможливість швидкої заміни

акумулятора, більшість систем подібного типу використовують два елементи як плюс та мінус через що у випадку помилки при посадці без запобіжних систем можна пошкодити акумулятор та всю електроніку дрону, а також не можна ігнорувати можливість пошкодження однієї з ніжок при посадці не на платформу та при жорсткій посадці.

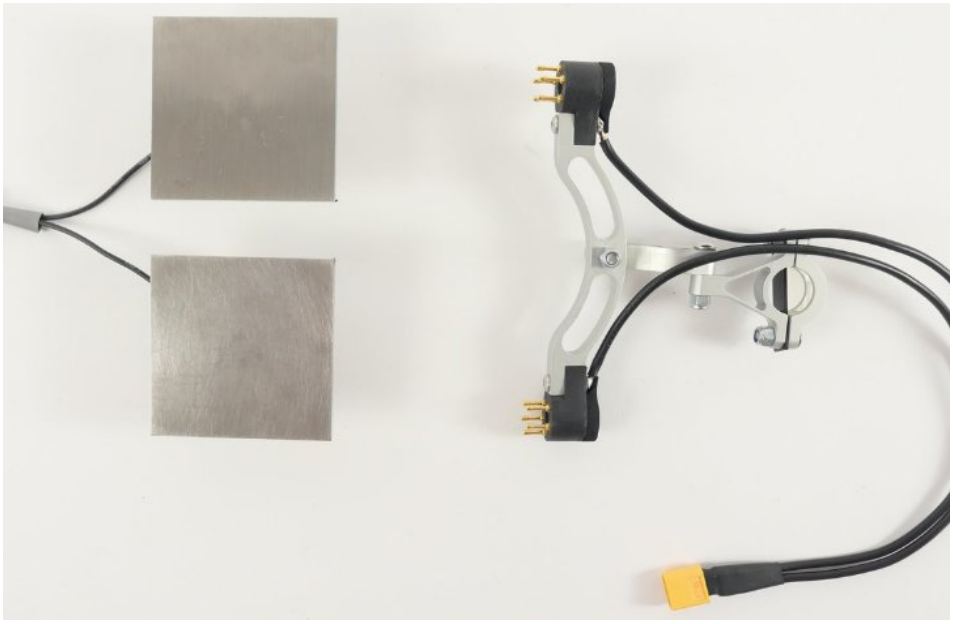

Рисунок 1.1 - SkyCharge з двома контактними платформами

Безконтактне (індукційне) заряджання передбачає використання електромагнітного поля для передачі енергії між станцією та дроном без прямого контакту. Метод зазвичай застосовується для безконтактної зарядки телефонів та показав себе надійним в подібній галузі. Такий підхід дозволяє не зосереджуватися на точності розміщення дрону на платформі, мінімізує знос контактів але має недоліки в швидкості зарядки яка набагато повільніше. WiBotic розробляють індукційні зарядні системи, які можуть адаптуватися до різних моделей дронів, головною вимогою є характеристики акумуляторів. Індукційна платформа складається з котушки на землі та приймальної котушки на дроні (рисунок 1.2).

Головними недоліком подібної системи є низький коефіцієнт корисної дії у порівнянні з дротовим заряджанням, висока вартість компонентів та вага приймальних котушок на дроні [9].

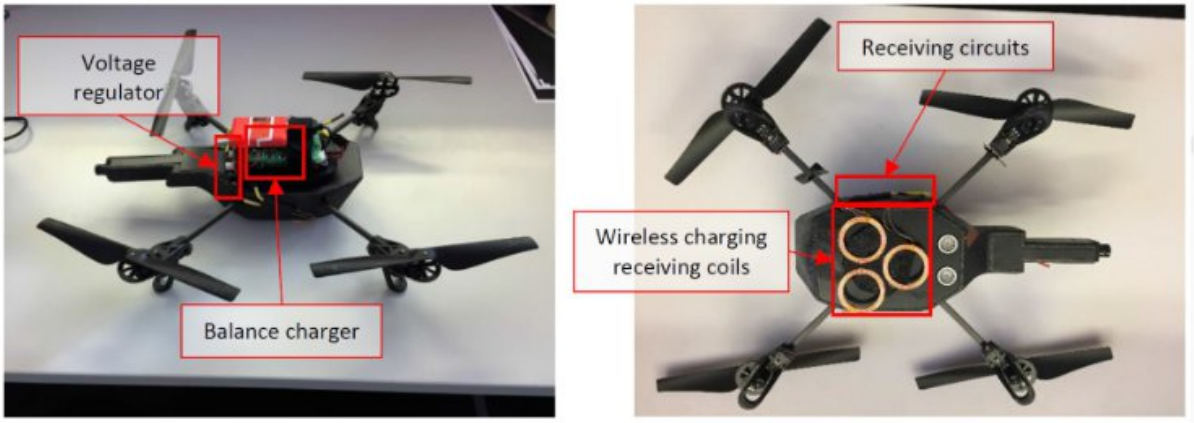

Рисунок 1.2 - Дрон WiBotic

Автоматизовані станції заміни акумуляторів це інноваційні станції які передбачають не лише заряджання, а й повну автоматичну заміну акумуляторів. Це дозволяє значно скоротити час не активності дрону. Замість очікування годин або кількох годин на зарядку, дрон отримує нову повністю заряджену батарею та продовжує виконання поставленого завдання. Одним з реалізованих прикладів є Airobotics DroneDock - система схожа на портові станції, яка дозволяє дрону сідати на платформу станції яка спускає дрон в невелику кімнату для виконання заміни, де робот маніпулятор витягує розряджений акумулятор і вставляє новий, поміщаючи розряджений акумулятор до слоту зарядки як зображено на рисунку 1.3.

Головним недоліком подібної системи є її висока вартість, оскільки потрібно реалізувати робота маніпулятора або купити та налаштувати під необхідні вимоги, але реалізація подібної станції заміни має величезну кількість переваг, головною з яких є мінімальні внесення змін в вагу та ціну дрону. Використання автоматизованої станції заміни акумуляторів також створює

потребу в стандартизації акумуляторів а також спрощенні та автоматизації механізму кріплення для зменшення витрат на робота маніпулятора.

Рисунок - 1.3 Airobotics DroneDock

Використання сонячних панелей для живлення дронів під час використання також є одним з важливих напрямів збільшення часу безперервної роботи. Подібний метод вимагає інсталяції кількох сонячних панелей, переважно обирають гнучкі через можливість надання контурів крила так і через нищу вагу, та інвертора для зарядки акумуляторів. Не дивлячись на потенційні переваги подібний метод зазвичай доцільно використовувати лише в літакоподібних дронах, оскільки для максимальної потужності сонячні панелі повинні бути направлені в одному напрямку до сонця та займати доволі велику площу, що легко можна реалізувати використовуючи крило. Як результат подібної інтеграції компанія XSun розробила дрон SX1-ISR (рисунок 1.4), який є повністю електронним дроном з вмонтованими дванадцятьма сонячними панелями які здатні під заряджати літальний апарат під час виконання завдань.

За заявами компанії дрон виконує місії на протязі чотирьох - семи годин спокійно, та теоретично здатен знаходитися в повітрі до двадцяти годин проводячи спостереження автоматизовано . Швидкість розрядки сильно залежить від навантаження на мотор, але якщо порівнювати стандартні квадрокоптери які здатні протриматися в повітрі годину це велика перевага.

Потрібно розуміти що подібне рішення не є ідеальним, оскільки гнучкі сонячні панелі не відрізняються великими потужностями. Дивлячись на розміри сонячних панелей вмонтованих у крила на моделі можна припустити що потужність сонячної панелі знаходиться в межах сотні ват, отже усі дев’ять панелей теоретично можуть видавати до дев'ятисот ват у найкращому випадку, але враховуючи кут нахилу крил та вигину сонячних панелей слід розуміти що показники не будуть ідеальними. Стандартний дрон швидко розряджає свою батарею через високе споживання електродвигуна, стандартні двигуни для квадрокоптерів працюють від 22 вольт та зазвичай тягнуть кілька ампер в залежності від навантаження, як результат витрата в понад двісті ватів для дрону вагою в кілька кілограм, звісно літакоподібний дрон має кращу аеродинаміку, що дозволяє дрону планувати протягом певного часу без використання моторів.

З документації XSun SX1-ISR можна дізнатися що його вага складає 35 кілограмів, для підйому подібного апарату потрібно створити тягу відповідну його вазі, тобто два мотори здатні підняти літак вагою в 35 кілограмів (ігноруючи пораду в збільшенні доступної ваги), при використанні двох моторів це складе 17,5 кілограмів на мотор, або 175 Н на один мотор.

Знайшовши мотори з необхідними характеристиками можна побачити, що подібні мотори споживають по 2500-3000 Вт, отже подібний дрон споживає 5000 - 6000 Вт, що сонячні панелі з загальною потужністю в 1000 Вт ніколи не можуть забезпечити для звичайного дрону, але оскільки дрон є літакоподібним, можемо припустити що при плануванні електроспоживання впаде до кількох

сотень ват, а при підтриманні крейсерської швидкості до двох тисяч ват, що потенційно дозволить подібному дрону спокійно літати. Звісно це лише теоретичні цифри, оскільки не були враховані аеродинамічні характеристики дрону та ігноровано якість повітряного потоку [10].

Рисунок 1.4 - XSun SX1-ISR

1.2 Огляд система підключення акумуляторної батареї

Одним із головних елементів у конструкції квадрокоптера чи будь-якого дрону є система підключення акумуляторної батареї. Від надійності цього з’єднання залежить стабільність роботи дрону і як результат загальна надійність польоту. Для автоматизації процесу заміни акумуляторної батареї слід розглянути основні типи конекторів, які використовуються у дронах, їхні особливості, переваги та недоліки.

Серед комерційних рішень найчастіше можна зустріти використання конекторів серії XT, AS, EC або унікальні контактні методи невеликих фірм.

XT-серія (XT90, XT60, XT30) (рисунок 1.5) - це найпоширеніша серія серед конекторів у сфері аматорських безпілотників. Цей тип конекторів

відомий своєю надійністю, підтриманням високого навантаження та зручністю під час монтажу [11].

XT30 застосовується у маленьких дронах вагою до кілограму та розрахований на струм до 30 ампер, що при стандартному живленні 22 В витримує 660 Вт, тобто двигуни в 150 Вт. XT60 - універсальний варіант для дронів середньої потужності, витримує до 60 ампер струму.

XT90 розроблений для важких систем та здатен працювати з навантаженнями до 90 ампер. Роз’єми XT мають плоскі контакти, позолочення та щільне з'єднання, яке запобігає випадковому від'єднанню.

Рисунок 1.5 - Конектор серії XT60

Серія EC (EC3, EC5) зображено на рисунку 1.6 має круглі контакти й дещо іншу форму корпусу. Застосовуються переважно у продукції виробника Horizon Hobby, а також знайшли поціновувачів у деяких професійних дронах. EC3 підходить для середнього класу дронів, EC5 для потужних систем до 120 А, що робить їх кращими за серію XT у випадках реалізації складних проектів. Їхньою перевагою є легкість підключення та висока площа контакту, хоча вони менш поширені в порівнянні з XT.

Рисунок 1.6 - Конектор серії EC 3

Серія AS (AS150, AS150U) максимально схожа за дизайном на серію XT, але має додаткові кріплення між стандартними. AS150 і AS150U застосовуються у промислових, аграрних та вантажних дронах, оскільки вони витримують рекомендоване навантаження 150 ампер але здатні працювати і при 180, забезпечують надійний контакт та мають вбудовані анти пікові резистори, які зменшують іскріння при з’єднанні. Це особливо важливо при використанні акумуляторів з високою напругою.

При використанні описаних вище конекторів також використовують балансувальні дроти (балансний конектор). Це окремий маленький роз’єм із кількома тонкими проводами - зазвичай чотири або п’ять. Де один провід - загальний мінус а всі інші йдуть від кожної серії (банки) акумулятора. Наприклад, у 3S акумулятора (11.1 В) буде одна головна пара EC для плюсу та мінусу та чотири дроти - земля та три на кожну банку (рисунок 1.7).

Балансування банок під час зарядки виконують для того, щоб кожна з банок мала однакову напругу та для моніторингу напруги по банках, якщо зарядний пристрій або дрон підтримує. Без балансування LiPo акумулятор швидко деградує або навіть вибухає, через що при створенні автоматизована система зарядки дронів слід враховувати необхідність додаткових балансувальних з’єднань при використанні стандартних акумуляторів.

Рисунок 1.7 - Акумулятор 3S

Подібний тип конекторів є надійним але вимагає зусиль для роз’єднання, що може створити проблеми при використанні роботу маніпулятору або іншого методу для від’єднання акумуляторної батареї, скільки потрібно прикласти зусилля в двох точках з’єднання.

Власні роз’єми виробників або унікальні роз’єми є найкращим вибором у випадку конкретної задачі. Багато виробників дронів використовують запатентовані роз’єми для батарей, які поєднують живлення, комунікаційні лінії, балансувальне з’єднання та механічне блокування в одному інтерфейсі, тобто системи за використанням схожі на акумуляторні батареї у версіях телефонів зі змінними акумуляторами, як приклад можна розглянути варіант DJI зображений на рисунку 1.8. Такі системи дозволяють вставити батарею в слот без додаткових дій та витягти її без створення навантаження на дрон, автоматично фіксуючи її та забезпечуючи надійне з’єднання що може виявитися зручним при виконанні нашої задачі.

Основним недоліком таких рішень є відсутність сумісності з іншими платформами, та необхідність в закритому корпусі через відкритість з’єднань,

що може призвести до пошкоджень або потрапляння сторонніх об'єктів у місце з’єднання та намокання [12,13].

Рисунок 1.8 - Змінний акумулятор DJI

1.3 Висновки до першого розділу

У ході огляду сучасних рішень для автоматизованої зарядки та заміни акумуляторних батарей дронів було проаналізовано основні існуючі технології, їх переваги та недоліки, а також оцінено доцільність їх застосування в умовах, наближених до реальних сценаріїв використання.

Основна увага була приділена чотирьом підходам до забезпечення безперервної роботи дронів: контактне заряджання, індукційне (безконтактне) заряджання, автоматизована заміна акумуляторів та використання сонячної енергії. Контактні зарядні системи, на зразок розробки компанії SkyCharge, продемонстрували високу ефективність та простоту реалізації, проте мають ряд суттєвих обмежень. Основним недоліком таких рішень є необхідність високої точності посадки дрону, що потребує додаткових систем позиціонування, а

також ризик виходу з ладу через механічні пошкодження контактних елементів. Індукційне заряджання, яке не вимагає фізичного контакту, забезпечує зручність та зменшує знос елементів, однак поступається контактному у швидкості зарядки. Цей метод вимагає встановлення додаткових компонентів (котушок), що збільшує вагу дрону та зменшує час його автономної роботи. Ефективність передачі енергії також нижча, що особливо критично при потребі швидкої підзарядки під час інтенсивної експлуатації. Використання сонячних панелей, як показав приклад дрону SX1-ISR від компанії XSun, також є цікавим рішенням, яке дозволяє значно продовжити час польоту без перерви. Проте це рішення є доцільним лише для дронів літако подібного типу, які завдяки своїй аеродинамічній конструкції мають можливість планування, а отже менше енергоспоживання яке дозволяє ігнорувати колосальне для подібних систем збільшення ваги.

Найперспективнішим з погляду автономності системи є підхід із повною автоматизованою заміною акумуляторних батарей. Такі системи, як Airobotics DroneDock, дозволяють повністю ігнорувати час очікування на зарядку використовуючи заготовлені акумулятори. Дрон автоматично приземляється на станцію, де маніпулятор витягує розряджену батарею та змінює її на заряджену. Незважаючи на високу ціну, автоматизовані станції заміни батарей мають значний потенціал в аграрному секторі, сфері безпеки, спостереженні за інфраструктурою та в багатьох інших галузях, де критичним є постійне виконання завдань без перерв. Особливо варто зазначити, що подібні рішення майже не потребують змін у конструкції дрону, якщо він уже адаптований під модульну заміну акумулятора, у випадку виконання автоматизації з нуля дрону потрібно створити захисну коробку для акумулятору, яку можна друкувати на 3Д-принтері, що одразу можна враховувати при створенні каркасу.

Окрему увагу в огляді приділено питанням вибору відповідних конекторів для забезпечення надійного електричного контакту при підключенні

та заміні акумуляторів. Серед доступних рішень найбільш поширеними є серії XT, EC та AS. Усі вони мають схожі характеристики та тип механічного кріплення. Для повноцінної реалізації системи автоматизованої заміни акумуляторів у дроні необхідно враховувати складність роз'єднання механічного кріплення, стандартизацію акумуляторних блоків, а також наявність балансувальних роз'ємів для контролю стану окремих елементів батареї які також потрібно від'єднувати та під'єднувати.

Важливо також зазначити, що при автоматизації процесу заміни виникає потреба в точному позиціонуванні акумулятора в слоті, щоб забезпечити надійне підключення силових та сигнальних роз'ємів.

З огляду на проведений аналіз, можна зробити висновок, що найбільш збалансованим рішенням для дронів, які використовуються в умовах регулярного навантаження, є саме автоматизована заміна акумулятора на заряджену. Це дозволяє зменшити час не активності дрону до мінімуму, усунути залежність від людського фактору та максимально ефективно використовувати безпілотне обладнання.

Для реалізації такого методу необхідно вирішити групу проблем, зокрема розробити механізм автоматичної заміни з високою точністю позиціонування, а також налаштування програмного забезпечення для координації всіх процесів.

2. ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЗАРЯДКИ ДРОНІВ

2.1 Аналіз системних компонентів автоматизованої станції зарядки дронів

Найпростішим і найдешевшим способом реалізації подібної системи є модифікована під наші потреби методика яка використовується в ігрових автоматах з клешнею для виграшу призу яка використовується в системі керування 3D принтером. Використовуючи групу сервоприводів та вакуумну присоску система зможе легко розмістити присоску над точкою у якій знаходиться дрон, відкрити нею кришку, прикріпитися та дістати акумуляторну батарею, перемістити присоску в точку для зарядки та помістити її в роз'єм для зарядки, після чого виконати зворотній процес, розмістивши присоску над зарядженою батареєю, прикріпитися до неї присоскою, перемістити її в слот для зарядки та помістити назад в слот для акумуляторної батареї дрону, виконавши заміну акумуляторної батареї на нову та закрити кришку.

Іншим варіантом реалізації може послужити модифікація для витягування батареї існуюча система Airobotics DroneDock у парі з доковими станціями. Причини для поєднання є дві, використання роборуки є надзвичайно дорогим та вимогливим до точного позиціонування у випадку відсутності інтеграції систем розпізнавання положення дрону та може створити додаткові вимоги до точності або збільшення робочої зони роборуки. Використовуючи поєднання існуючих інженерних рішень DJI Dock 3 для точної посадки та створення систем маніпулювання за функціоналом схожої на Airobotics DroneDock які використовують спрощену систему роборуки використовуючи три сервоприводу для покриття максимального простору [14,15,16,17].

Отже за основу буде розглянуто дизайн DJI Dock 3 для позиціонування дрону та реалізувати спрощений робот маніпулятор використовуючи дешеву та просту в керуванні плату Arduino. Таким чином наша система може бути реалізована за допомогою групи сервоприводів, насосу-помпи у випадку якщо буде вирішено використовувати його а не робо клешнею у стандартній збірці, контролера Arduino, блоку живлення, та посадочної платформи з відкритими зарядними отворами, що зробить обраний спосіб зарядки порівняно дешевою та простою в обслуговуванні альтернативою існуючим системам.

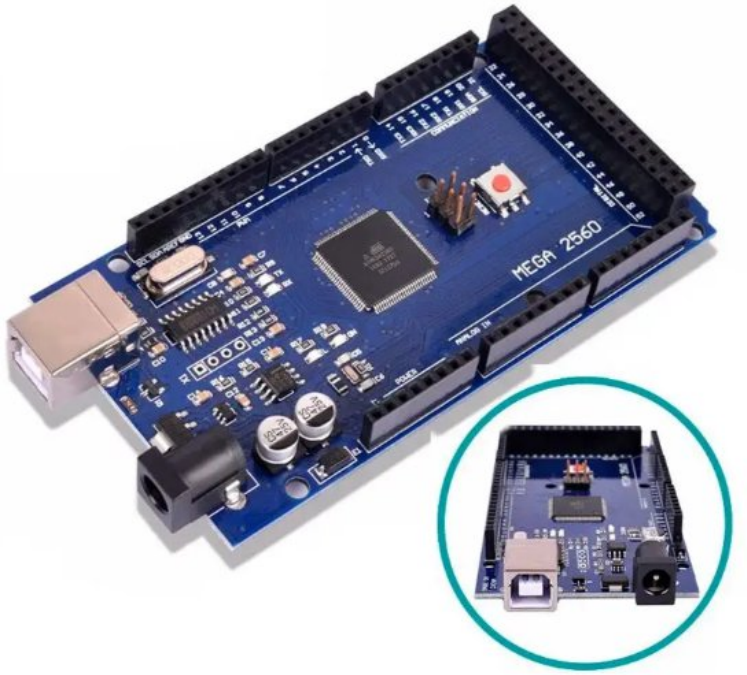

Рисунок 2.1 - Arduino Mega

У якості контролеру вирішено використовувати Arduino Mega (рисунок 2.1), однією з головних причин для такого вибору є її здатність забезпечувати достатню кількість входів і виходів які необхідні для реалізації нашої складної системи керування, що керує кількома сервоприводами та можливістю використання датчиків та сервоприводів, а також причиною вибору стала його наявність, хоча проект спокійно можна реалізувати використовуючи Arduino

Uno. Порівнюючи з Arduino Uno, плата Arduino Mega має значно потужніші характеристики, головна відмінність - Arduino Uno має тільки 14 цифрових пінів, входів/виходів, 6 аналогових пінів входів та 1 UART, тоді як Mega має в своїй архітектурі 54 цифрових піни входи/виходи, 16 аналогових пінів входів та 4 UART (рисунок 2.2), що забезпечує можливість модифікації приладу у майбутньому для інтеграції датчиків, хоча теоретично прилад буде використовувати не більше восьми пінів у стартовій версії, повні характеристики відображено в таблиці 2.1.

Це робить Arduino Mega кращим вибором для потенційного ускладнення проекту особливо при необхідності використання дисплею [18,19,20].

Таблиця 2.1 - Характеристики Arduino Mega

Характеристика	Значення
Мікроконтролер	ATmega2560
Тактова частота	16 МГц
ОЗУ (SRAM)	8-16 КБ
EEPROM	4-8 КБ
ПЗУ (Flash)	256 КБ (з яких 8 КБ виділено на завантажувач)
Цифрові входи/виходи (I/O)	54
Аналогові входи	16
UART (послідовні порти)	4
Інтерфейси	I2C, UART, SPI

Кінець таблиці 2.1

Робоча напруга	5 В
Вхідна напруга (рекоменд.)	7–12 В
Максимальна вхідна напруга	6–20 В
Струм на пін I/O	20 мА
USB-інтерфейс	Type B

Arduino Mega дозволяє системі ефективно розподіляти функціональність між підсистемами, що підвищує стабільність та гнучкість роботи всієї плати. Варіантом використання є застосування одного з UART-портів для зв’язку з модулем бездротової передачі даних для контролю приладом з телефону або комп’ютеру, інший для взаємодії з додатковим контролером, одноплатним або звичайним комп’ютером. Це особливо зручно під час налаштування системи чи проведення діагностики під час якої проводиться запис даних роботи елементів в реальному часі. Велика кількість доступних пінів дозволяє уникнути використання додаткових мультиплексорів, зсувних регістрів чи інших методів для збільшення доступних пінів керування, що сильно спрощує схему підключення та зменшує ймовірність помилок при під’єднанні чи внесені змінних як в код так і в схему підключення.

Ще однією перевагою є потужна підтримка з боку спільноти розробників та велика кількість інструкцій та навчальної інформації, що дозволяє швидко під’єднати нові компоненти, такі як датчики відстані, датчики вологості, насос та інші. Простота програмування в середовищі Arduino IDE також дає змогу спростити процес та скоротити час розробки а також надає змогу промоделювати програму, що забезпечити швидке тестування версій. У

розглянутому випадку це дозволило перевіряти ефективність та точність алгоритмів керування [21,22].

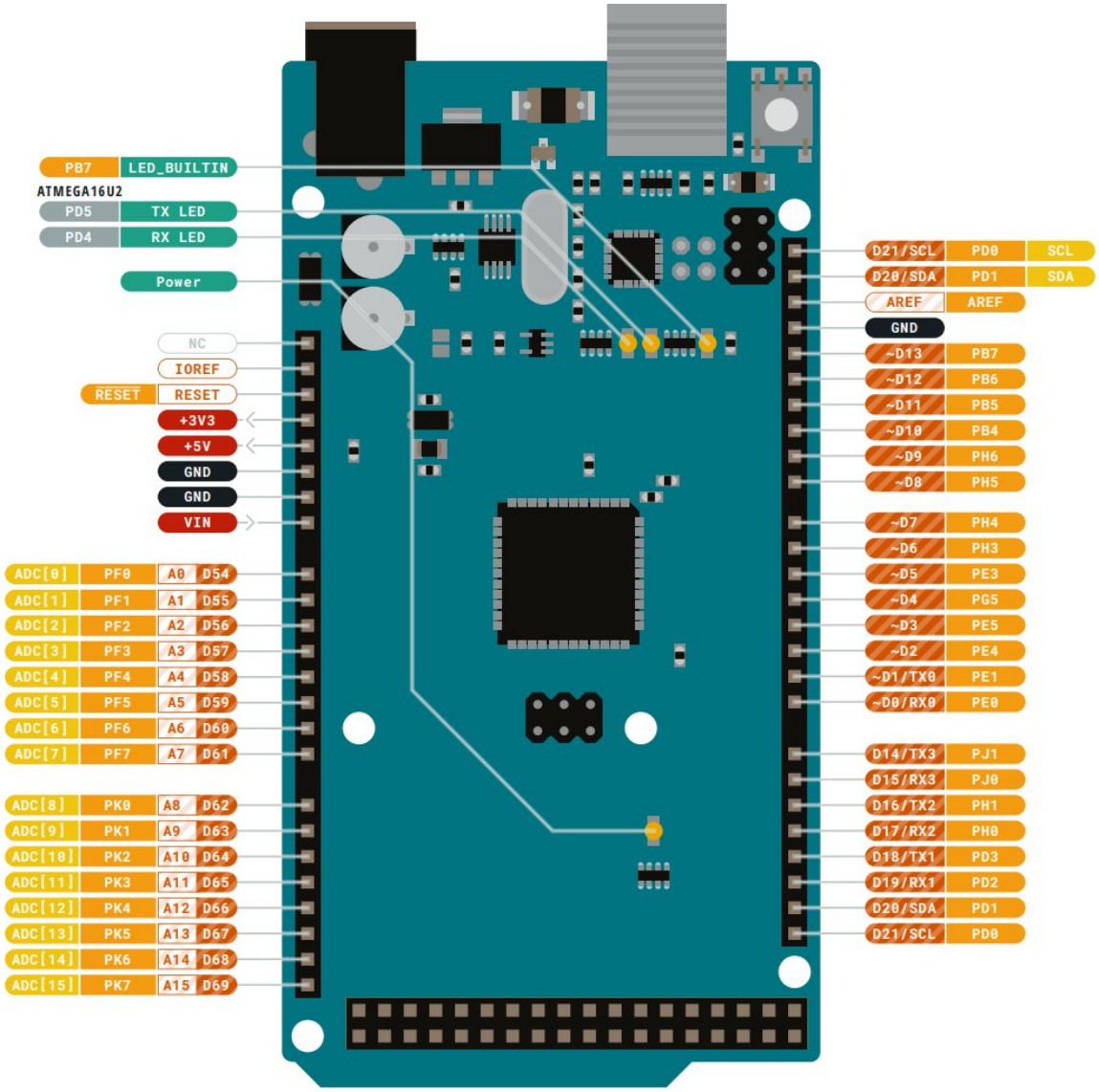

Рисунок 2.2 - Розпіновка Arduino Mega

Таким чином, вибір саме серії контролерів Arduino Mega при врахуванні можливої модифікації або UNO став обґрунтованим рішенням, яке забезпечило баланс між ціною, технічними можливостями та зручністю реалізації.

Живлення платою можна реалізувати через стандартний порт живлення використовуючи рекомендований виробником адаптер JB-0520 (рисунок 2.3),

який видає 5 В та 2 А що ідеально підходить для наших потреб, оскільки Arduino Mega маэ робочу напругу в 5 В та граничну в 11 В, хоча шанс пошкоджень в певній документації розпочинається при 7 В.

Рисунок 2.3 -Імпульсний адаптер живлення Voltronic

Рисунок 2.4 - Сервопривід MG996R

Для реалізації руху буде використовуватися три сервоприводи MG996R (рисунок 2.4). Які будуть встановлені в основі приладу для обертання платформи відносно точки встановлення, та в двох ліктях робота маніпулятору для покриття максимальної площі роботи системи. Сервопривід MG996R було обрано у зв’язку з його оптимальному поєднанню потужності, надійності та

доступності у магазинах. Порівнюючи з менш потужними аналогами, MG996R забезпечує високий обертовий момент до 9.4 кг/см при напрузі 5 В. Це критично важливо для маніпулятора, оскільки такий момент дозволяє рухати вагу нашого приладу та додатково піднімати й утримувати акумуляторну батарею яка може мати вагу в кілька кілограмів без втрати точності, зупинок у русі чи ривків при високому навантаженні.

Другою причиною стала сумісність із Arduino Mega / UNO. MG996R легко підключається до цифрових пінів мікроконтролера та легко контролюють за допомогою існуючої стандартної бібліотеки Servo.h, що дозволить реалізувати прості процеси за лічені секунди. Це спрощує процес програмування та як результат процес налагодження системи, що особливо важливо на етапі прототипування та тестування.

Окрім цього, MG996R має металеві шестерні, що суттєво підвищує його зносостійкість порівняно з моделями які надають перевагу пластику. Це дозволяє використовувати MG996R без внесення змін протягом довгого періоду часу та забезпечує стабільну роботу маніпулятора.

MG996R - це сервопривід, який обертає свій вал на певний кут, зазвичай від 0 до 180 градусів що створює певні обмеження через обмеження в зоні досягу, але для нашої мети цього досить, та завдяки правильному позиціонуванні система зможе мінімізувати втрати та обмеження які створить доступний кут роботи системи. Усередині сервоприводу є маленький мотор, редуктор (система шестерень, які відповідають за зменшення швидкості, для збільшують сили), потенціометр (відповідає за вимір кута в момент роботи) та електронна плата яка всім керує.

Контроллер Arduino надсилає сигнал на сервопривід. Тривалість цього сигналу вказує кут на який потрібно повернути вал. Для прикладу якщо імпульс триває 3 мс, то сервопривід ставить вал приблизно на 180 градусів. Якщо менше кут відповідно менше.

MG996R постійно запитує та порівнює, у якій позиції він перебуває, та в яку позицію йому треба ставитися. Потім зупиняється й утримує це положення, поки не надійде нова команда.

Для більш точних дій буде використовуватися сервопривід TowerPro SG90 (рисунок 2.5) схожий за принципами роботи на MG996R .

SG90 Micro Servo - це маленький і легкий сервопривід, який часто застосовують у простих проектах, де не потрібно великого зусилля та слід мінімізувати вагу пристрою. SG90 Micro Servo важить 9 грамів, споживає мало струму, і працює з напругою 4.8 - 6 В, що в п’ять разів слабше за MG996R, але система буде використовувати його в частині робо клешні або взагалі не буде використовувати у випадку якщо при розробці буде обрано використовувати версію з присоскою, але через відсутність руху усім приладом а лише об’єктом над яким виконуються маніпуляція цей сервопривід ідеально підходить для виконання поставлених задач при необхідності використання у парі з Arduino [23-29].

У порівнянні з MG996R який значно більший(4 х 2 х 4.2 см) та важить 50 г, цей елемент займає лише 2.3 х 1.2 х 2.9 см та важить 9 г. Різниця в характеристиках призводить до того що SG90 це малий в розмірах, слабкий та дешевий сервопривід який підходить для легких завдань. Якщо навантаження велике цей сервопривід починає дзижчати або проскакує втрачаючи точність вимірів що може призвести до проблем якщо необхідна точність. MG996R це великий, потужний, дорожчий сервопривід який витримує вагу, краще тримає позицію при навантаженні(у нашому випадку це вага конструкції робота маніпулятору та акумуляторної батареї при взаємодії), не зношується так швидко під навантаженням.

Рисунок 2.5 - Сервопривід TowerPro SG90

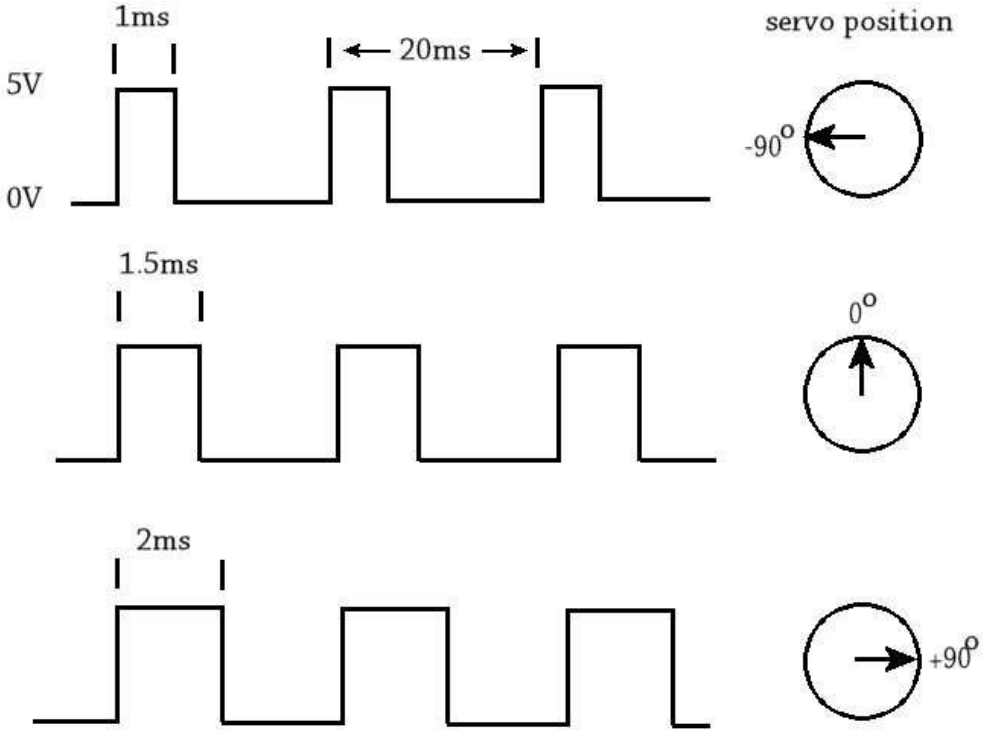

Рисунок 2.6 - PWM контроль сервоприводу

Обидва сервоприводи контролюються за допомогою контролеру Arduino. Сервопривід SG90 чи MG996R керується за допомогою широтно-імпульсної модуляції (PWM). Обраний нами контролер (Arduino Mega) надсилає групи

імпульсів, які відповідають за кут повороту валу сервоприводу. Частота сигналу 50 Гц з періодом 20 мс (рисунок 2.6). Тривалість імпульсу визначає кут в залежності від встановлення приладу:

- 1 мс – приблизно 0°
- 1.5 мс – приблизно 90°
- 2 мс – приблизно 180°

Принцип роботи наступний, контролер надсилає постійно повторювані імпульси, SG90 або інший сервопривід зчитує тривалість імпульсу й обертає вал на відповідний кут (система визначає довжину сигналу в перервах між ними в 20мс, тобто сигнал 1.5мс говорить сервоприводам стати в позицію 90 градусів, вирівнятися в центр доступного діапазону, після чого йде очікування в залишок 20 мс і повтор доки йде сигнал для підтвердження та утримання положення). Внутрішній потенціометр дозволяє сервоприводу SG90 утримувати задану позицію, поки не зміниться імпульс, при зникненні імпульсу прилад продовжує утримувати позицію наче вимкнули живлення [31,32,33].

Для реалізації можливості дистанційного керування та запису даних обрано використовувати модуль SPP-C Bluetooth із адаптером SPPC HC-05 HC-06 для Arduino (рисунок 2.7). Модуль HC-05 і HC-06 - це найпопулярніші Bluetooth модулі, які дозволяють Arduino обмінюватися даними зі смартфоном, комп’ютером або іншим пристроєм який підтримує Bluetooth. Вони від’єднуються через послідовний порт (UART) і їх можна безпечно підключати напряму до Arduino, яке працює на 5 В.

Різниця між HC-05 і HC-06:

- HC-05 можна налаштувати як майстер або підлеглий, він може ініціювати з'єднання з іншим Bluetooth-пристроєм або чекати, поки з'єднаються з ним;
- HC-06 працює тільки в режимі підлеглий і він лише приймає сигнал

На телефоні створюється додаток "Bluetooth Terminal". Коли встановлено з’єднання, усе, що надсилається із телефону, приймає Arduino через Serial, і навпаки.

У ситуації якщо при реалізації буде обрано використовувати присоску для від’єднання батареї та відкриття захисної кришки можна використовувати модуль з використанням вакуумного насосу чи повітряної помпи air pump (рисунок 2.8), клапан (опційно) та MOSFET або реле.

Вакуумний насос виконує відкачування повітря з певної порожнини комірки (в нашому випадку з простору під присоскою). Коли повітря відкачано, тиск під присоскою стає менший за атмосферний і як результат атмосферний тиск притискає присоску до поверхні об’єкта. Для нашої задачі підійде любий прилад здатний підняти 0.5 кг які буде важити велика акумуляторна батарея, у випадку необхідності модифікації для використання більш важкої батареї можена виконати зміну та використати іншу повітряну помпу.

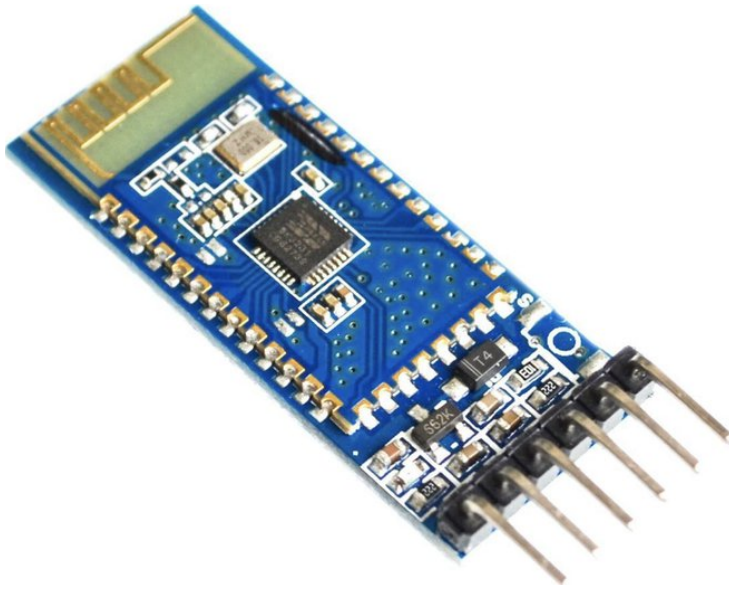

Рисунок 2.7 - Модуль SPP-C Bluetooth із адаптером SPPC HC-05

Елементом використання якого є під питанням через його не критичність для нашої системи є електромагнітний клапан. Без клапана насос просто качає

повітря створює вакуум у присосці, та прилипає, але як тільки система вимкне насос, тиск через щілини починає повертатися назад і вирівнюватися з тиском атмосферним і вакуум поступово зникає.

Клапан в свою чергу дозволяє зафіксувати положення та підтримувати тиск протягом більшого часу.

Отже якщо не встановлювати клапан буде отримано наступний результат при використанні лише насосів:

- слабке утримання при вимкненому насосі;
- потрібно постійно тримати насос увімкненим;
- витрати більшої кількості енергії;
- швидше зношування насосу через постійну роботу.

Якщо система буде використовувати клапан то будуть отримаго наступні переваги:

- вакуум зберігається без живлення насоса;
- можна відключити насос;
- енергоефективність і надійність.

Рисунок. 2.8 - Повітряна помпа

На перший погляд використання клапану є найкращим варіантом але проблемою виступає його ціна та керування у більшості виконується через 12В, для прикладу XH-W3001 Mini Solenoid Valve який підходить для нашого проекту керуються через 12В, що змусить нас використовувати додатковий транзистор (типу MOSFET) або модуль реле.

MOSFET (типу IRF520 або IRLZ44N) - це в певному сенсі аналог реле, електронний вимикач. Arduino не може напряму вмикати мотор чи насос, бо не витримує великого струму в 12В. Замість цього у системі використовуватимемо MOSFET, Arduino дає низький сигнал на керуючий пін MOSFET, MOSFET пропускає великий струм від джерела живлення через насос до землі що запускає насос чи помпу та створює вакуум.Коли сигнал з Arduino зникає MOSFET вимикає ланцюг, та насос зупиняється і клапан відчиняється.

Рисунок 2.9 - Трьох роз'ємний зарядний модуль для DJI Mini

У випадку використання насосу чи клапану при реалізації з’являються нові вимоги пов’язані з використанням зовнішнього живлення, подібна вимога не виникає при використанні лише сервоприводів, тому при моделювання слід

врахувати цей момент або спробувати найти компоненти які б не потребували подібних модифікацій.

Головною причиною чому в системі не можна під’єднати насос напряму до ардуіно або використати перетворювачі є 5В та максимум 500мА які здатні підтримувати плата на всіх пінах, як результат при спробі під’єднати насос який споживає від 200 - 500 мА при 12 В система повинна збільшити споживання до 400 - 1000 мА, що в свою чергу може призвести до критичного пошкодження плату чи пінів.

Найважливішим елементом нашої системи є блок живлення - зарядки акумуляторних батарей, нами після аналізу існуючих було обрано використовувати трьох роз'ємний зарядний модуль для DJI Mini зображений на рисунку 2.9. який не має вимог до встановлення чи використання.

Завдяки його простій структурі оператор з легкістю зможе за допомогою робота маніпулятора відновити акумуляторну батарею в слот для заряду відпустивши його після позиціонування, а після завершення схопити акумулятор та перенести його в роз’єм дрону.

2.2 Розробка системи утримання об’єкту

Як було обумовлено в попередньому розділі у системі буде використовуватися два типи захоплення, робо клешню або присоску, які будуть позиціонуватися за допомогою робо руки (рисунок 2.10) [35,36].

Контроль захопленням об’єкту буде виконуватися у точці Gripper за допомогою SG90, який буде приводити в рух дві пластикові шестерні, які передають цей рух в робо клешню.

Можна приблизно розрахувати, яку максимальну вагу здатний утримувати SG90 у типовій конфігурації з робо клешнею за формулою:

$$F = T/r, \quad (2.1)$$

де: F- сила, яку може створити сервомотор (Н, або кг * 9.81);

T - крутний момент сервоприводу (Н*м);

r - відстань від осі обертання до центру маси вантажу (м).

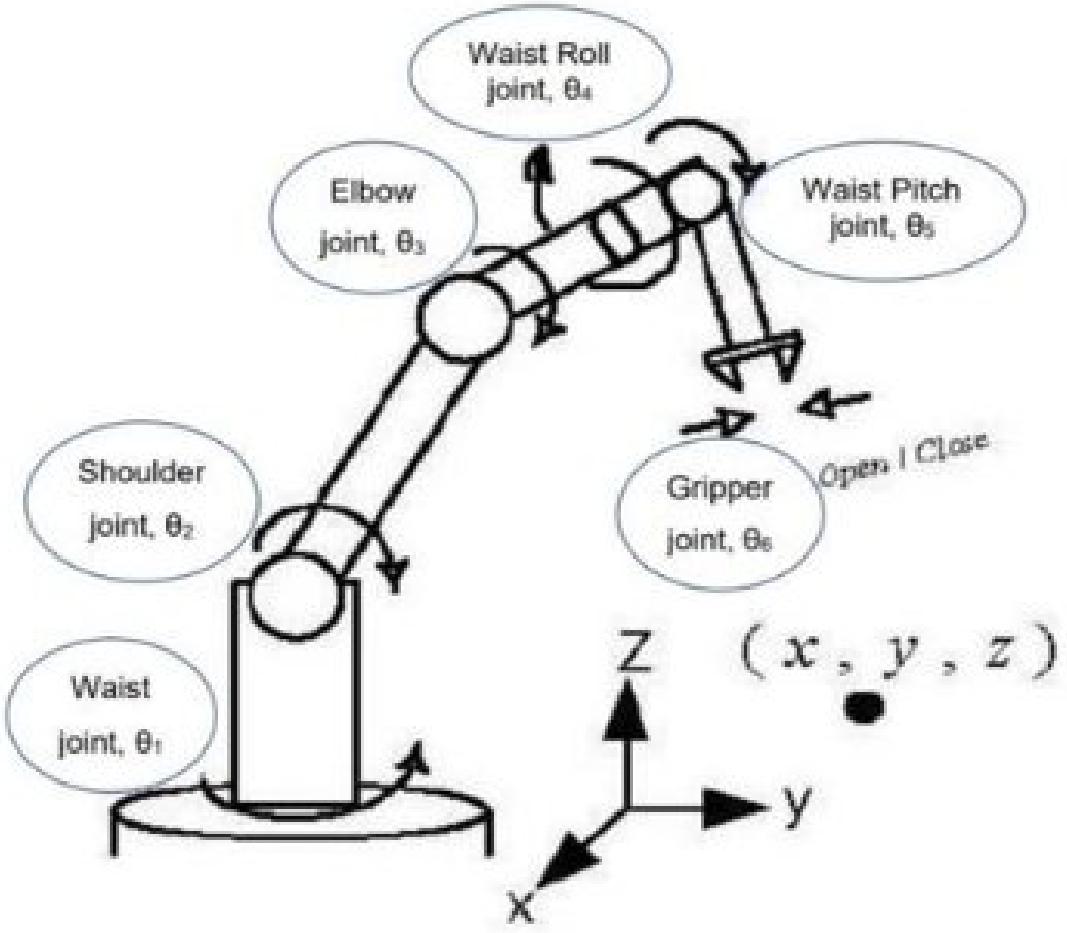

Рисунок 2.10 - Схема робо руки

При використані обумовленого SG90 у якого крутний момент сервоприводу складає 1.2 кг*см, тобто 0.18 Н*м та ширину батареї в 3 см після підставленні усіх параметрів можна отримати рівняння 2.2:

$$F = 0.18\ \text{Н} * \text{м} / 0.03\ \text{м} = 6\ \text{Н};\qquad (2.2)$$

Отже цей сервопривід при прямому застосуванні у конструкції з двома пальцями, з'єднаними через шестерні (одна ведуча до якої приєднано сервопривід, а інша ведена) здатен піймати 6 Н або 0.6 кг, чого достатньо для нашого завдання, але можливо покращити це число зробивши зміни в шестерні використовуючи передачу з пониженням використовуючи формулу коефіцієнта передачі(редукції):

$$i = Z_{\text{ведена}} / Z_{\text{ведуча}};\qquad (2.3)$$

i - коефіцієнт передачі (редукції);

$Z_{\text{ведена}}$ - кількість зубів веденої шестерні;

$Z_{\text{ведуча}}$ - кількість зубів ведучої шестерні.

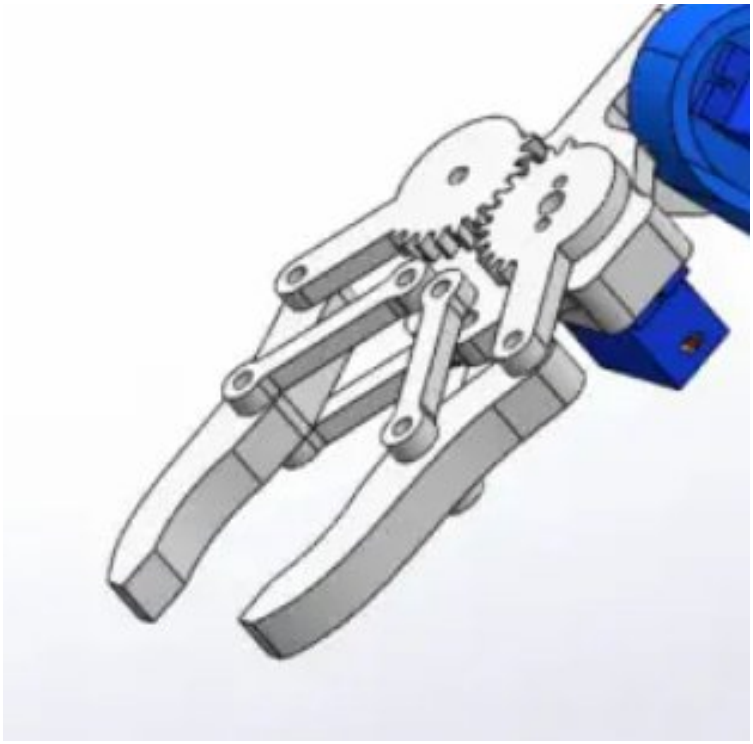

Рисунок 2.11 - Модель робо клешні

Якщо $T_{\text{серво}}$ - крутний момент сервопривода, то крутний момент на веденій шестерні визначається наступним чином:

$$T = T_{\text{серво}} * i; \tag{2.4}$$

$$T = T_{\text{серво}} * Z_{\text{ведена}} / Z_{\text{ведуча}}. \tag{2.5}$$

Отже змінивши кількість зубців з 10 до 30 можна отримати наступний результат:

$$F = (\frac{T_{\text{серво}} * Z_{\text{ведена}}}{Z_{\text{ведуча}}})/r = (\frac{0.18 * 30}{10})/0.03 = 18\ \text{Н} \tag{2.6}$$

Як результат вносячи зміни в довжину клешень, відстань від осі обертання до центру маси, та внесенні змін в характеристики шестерень отримуємо можливість підіймати 1.8 кг, що створить навантаження на сервопривід, але теоретично система не повинна дійти до подібних навантажень. Остаточний дизайн захоплювача робота маніпулятору зображено на рисунку 2.11.

При використанні другого дизайну, а саме присоски в системі заміняється сервопривід у клешні на присоску яку можна зафіксувати у корпусі клешні для кращої фіксації або розробити замінний модуль, насос можна розмістити в основі приладу, таким чином зменшивши вагу існуючої клешні та використовувати рух з більшою доступною вагою, або розмістити в роз'ємі для SG90.

Вирахувати вагу об'єкта з яким система може взаємодіяти за допомогою присоски вираховується за формулою:

$$F = \Delta P * A; \quad (2.7)$$

F - сила всмоктування (Н);

ΔP - перепад тиску (Па) між атмосферним і внутрішнім (тобто вакуумом);

A - площа присоски (м²).

Площа присоски, A можна вивести за формулою:

$$\mathrm{A} = \frac{\pi * d^2}{4}. \quad (2.8)$$

Підставивши параметри нашого повітряного клапану отримаємо наступний результат:

$$F = 60000\ \text{Па} * \frac{\pi * 0.03\text{м}^2}{4} = 42.42\ \text{Н}; \quad (2.9)$$

$$m = 42.42 / 9.82 = 4.32\ \text{кг}. \quad (2.10)$$

Розрахунки показують що наша система з використанням вакуумної присоски здатна піднімати вагу до 4.32 кг, але в дійсності через не ідеальний контакт з поверхнею акумуляторної батареї та можливість її забруднень (хоча в більшості випадків вона буде у захисному корпусі), фактичне навантаження краще буде обмежувати до 50%, тобто 2.16 кг.

Якщо поверхня пориста або крива сила падає дуже сильно, тому слід враховувати це при виборі розміру та типу акумуляторної батареї.

2.3 Розробка структурної та схеми автоматизованої системи зарядки дронів.

Наша станція зарядки для дронів буде виглядати за основним дизайном схоже на докові станцій, у головному корпусі буде розташовуватися зібраний нами робот маніпулятор, зарядка (зарядні слоти) та місце для приземлення зображені на рисунку 2.12.

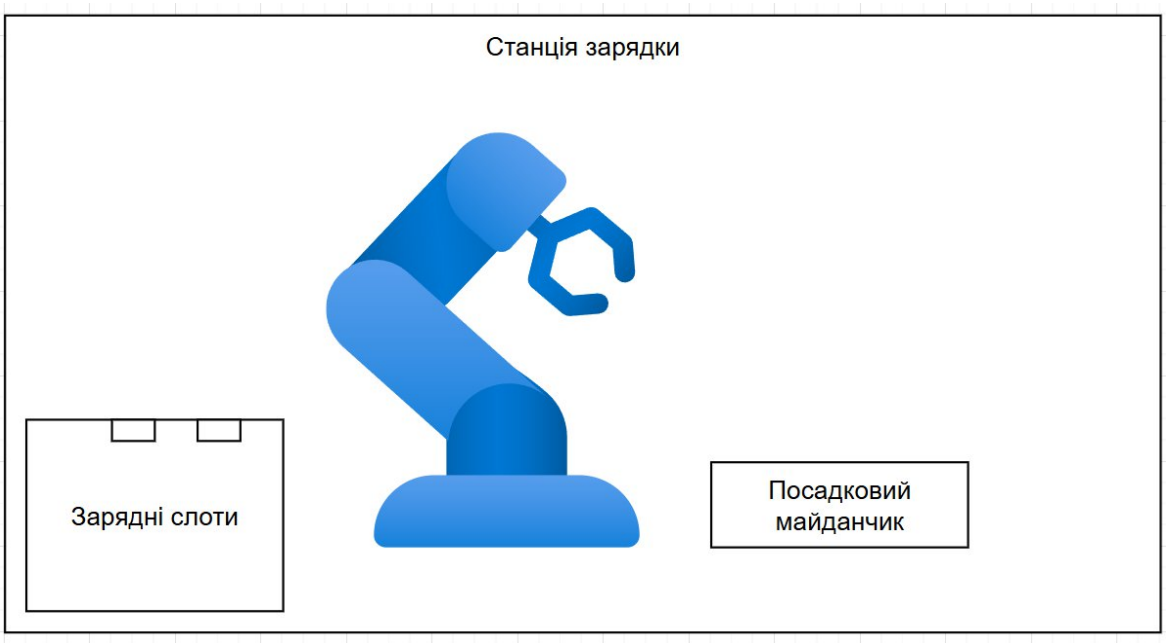

Рисунок 2.12 - Схематична схема системи зарядки дронів

На рисунку 2.13 зображено узагальнена структурна схема роботи системи з елементами (дроном та акумуляторами). Дрон приземляється на посадкове місце, акумуляторна батарея розташовується зверху дрону що робить її доступною для відкриття ти демонтажу.

Маніпулятор одним з двох можливих способів присоскою або клешнею відчинить кришку, захопить батарею горизонтально, витягне батарею горизонтально на двадцять сантиметрів щоб звільнити акумуляторну батарею з роз’єму аби мати місце при здійсненні оберту. Вільну акумуляторну батарею

буде повернуто на 90 градусів для розміщення вертикально для поміщення в зарядний роз’єм слоти в яких також розташовано вертикально (рисунок 2.14).

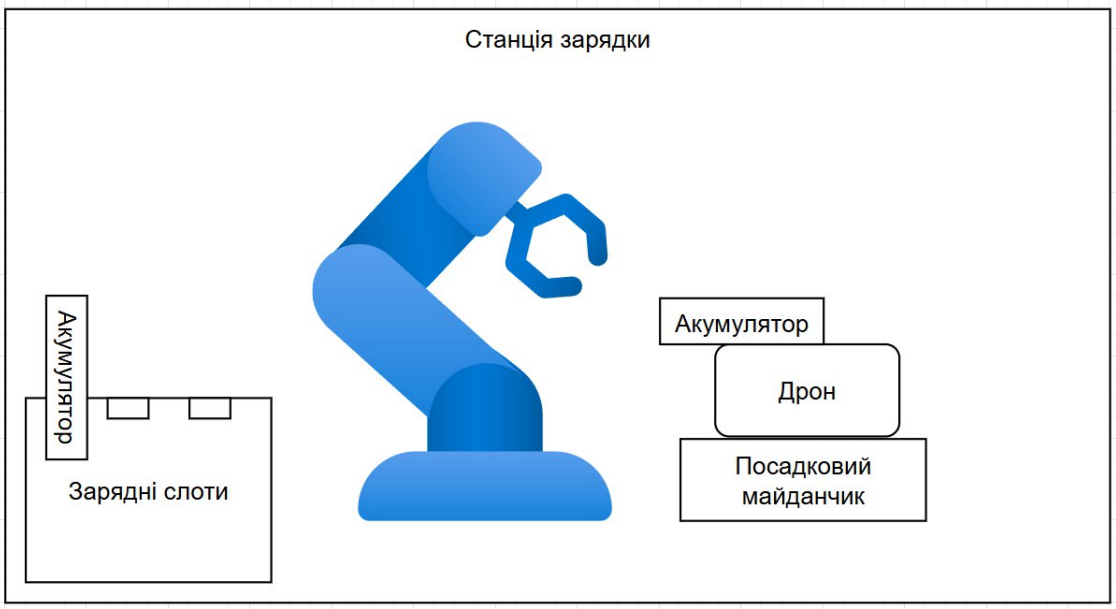

Рисунок 2.13 - Схематична схема системи зарядки дронів з елементами

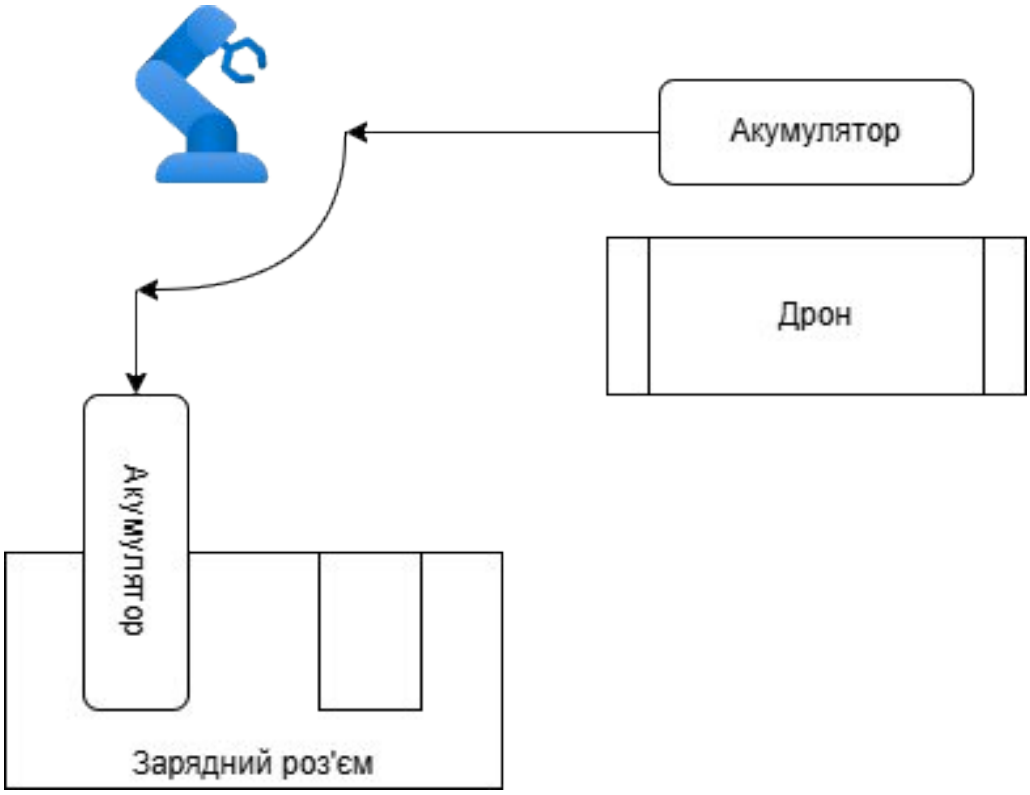

Рисунок 2.14 - Принцип переміщення акумуляторної батареї

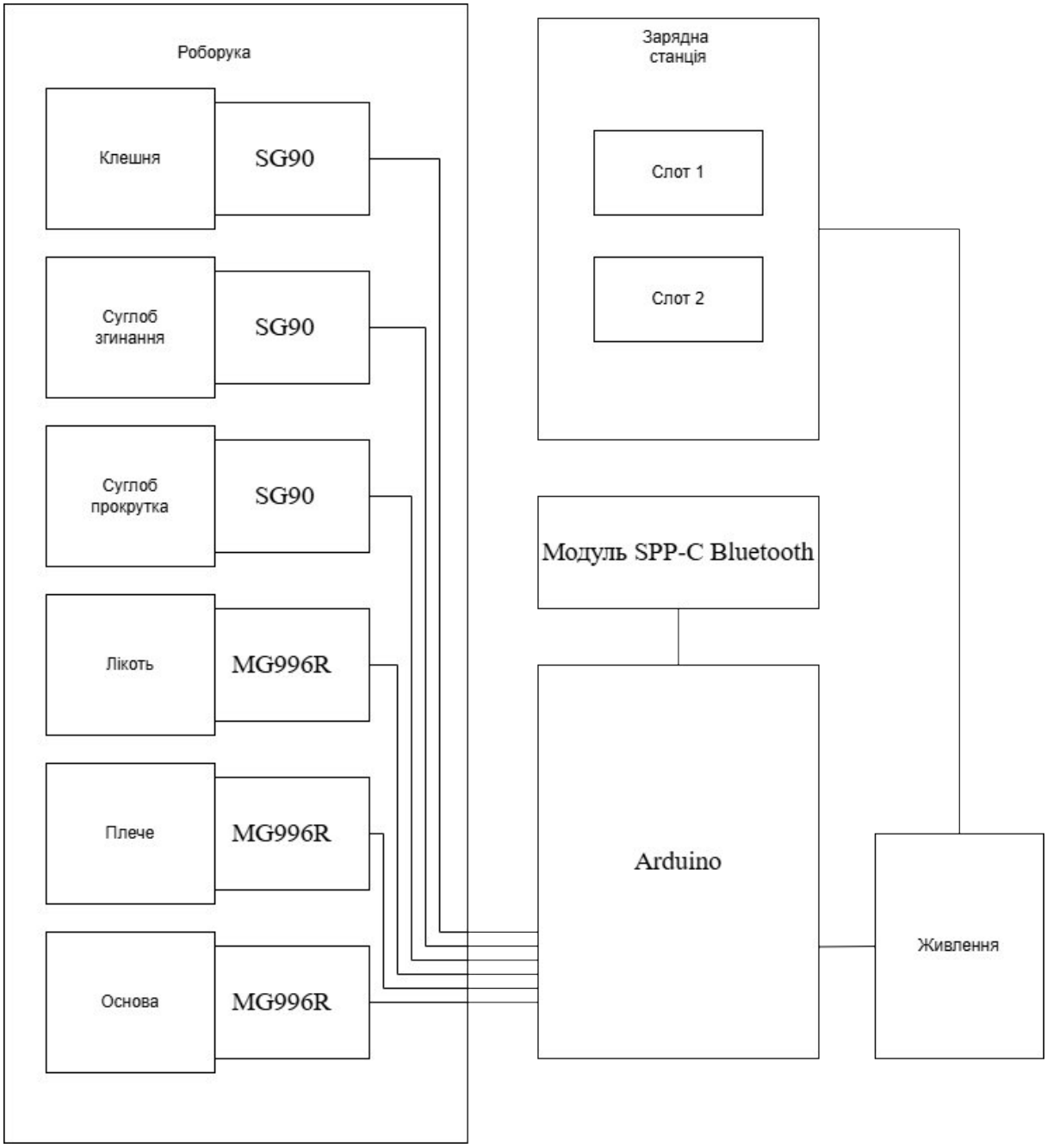

Рисунок 2.15 - Структурна схема автоматизованої системи зарядки дронів

Структурна схема робота маніпулятору автоматизованої системи зарядки дронів зображено на рисунку 2.15, як видно з рисунку наші сервоприводи будуть розташовані в роботі маніпуляторі та приєднуватися напряму до ардуіно який розташовується зовні або на основі робота маніпулятору або у захисному

корпусі, живлення подається через вхідний порт ARDUINO від електромережі. Зарядна станція розташовується біля роботу маніпулятору та живиться від зовнішньої мережі, як результат немає необхідності контролювати роз'єми з плати, а лише зберігати інформацію про стан кожного роз'єму та який з них вільний. Додатковий модуль для керування також живиться від ARDUINO та використовується для контролю при необхідності [37,38].

2.4 Висновки до другого розділу

У цьому розділі було обґрунтовано вибір компонентів застосованих для реалізації системи автоматизованої зарядки дронів шляхом автоматизованої заміни акумуляторної батареї та взаємодії з зарядною станцією.

Найбільш економічно та технологічно вигідним варіантом виявилася система, що використовує руку маніпулятора з присоскою або клешнею в залежності від доступних елементів та типу батареї та можливості внесення додаткових модифікацій в структуру дрону та акумуляторної батареї. Таке рішення дозволяє переконатися в забезпеченні потрібної точності, надійності та функціональність при мінімальних витратах.

Вибір контролера Arduino Mega, було зумовлено її технічними перевагами над Arduino Uno, це відкриває шлях до подальшого модифікування та інтеграції нових функцій без значної модифікації основної конструкції.

Для реалізації руху маніпулятора обрано сервоприводи MG996R за їх оптимальне співвідношення потужності, надійності та вартості. Додатково використання TowerPro SG90 дозволяє зменшити навантаження у другорядних елементах, де не потрібна велика сила.

З метою дистанційного керування нами було обрано Bluetooth-модуль HC-05, який дозволяє реалізувати просте і надійне з'єднання з пристроєм який має Bluetooth.

Система є відкритою до вдосконалення, здатна підтримувати модульний підхід і може бути адаптована до різних сценаріїв експлуатації в залежності від вимог шляхом заміни сервоприводів розташованих в клешні на інший елемент для утримання.

3. ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ЗАРЯДКИ ДРОНІВ

3.1 Підключення елементів роботу маніпулятору

Обрані нами сервоприводи MG996 та SG90 мають просту будову та використовують три входи для підключення, червоний - живлення +5В, коричневий відповідає за землю GND, та оранжевий який відповідає за отримання сигналів керування (рисунок 3.1).

Рисунок 3.1 - Входи сервоприводу

Підключення виконується напряму до плати, оскільки система використовуємо шість однакових сервоприводів можна використовувати драйвер але наша плата має достатню кількість виходів, через що під’єднуватися можна напряму через плату і її до неї. Найпростішим варіантом є під’єднання кабелю сервоприводу +5В під’єднати до виходу +5В плати, точно те саме виконати з землею під’єднавши до GND, ця дія повторюється для

кожного з елементів. Останній не під’єднаний оранжевий кабель PWM слід під’єднати до любого з пінів для передачі сигналу (рисунок 3.2).

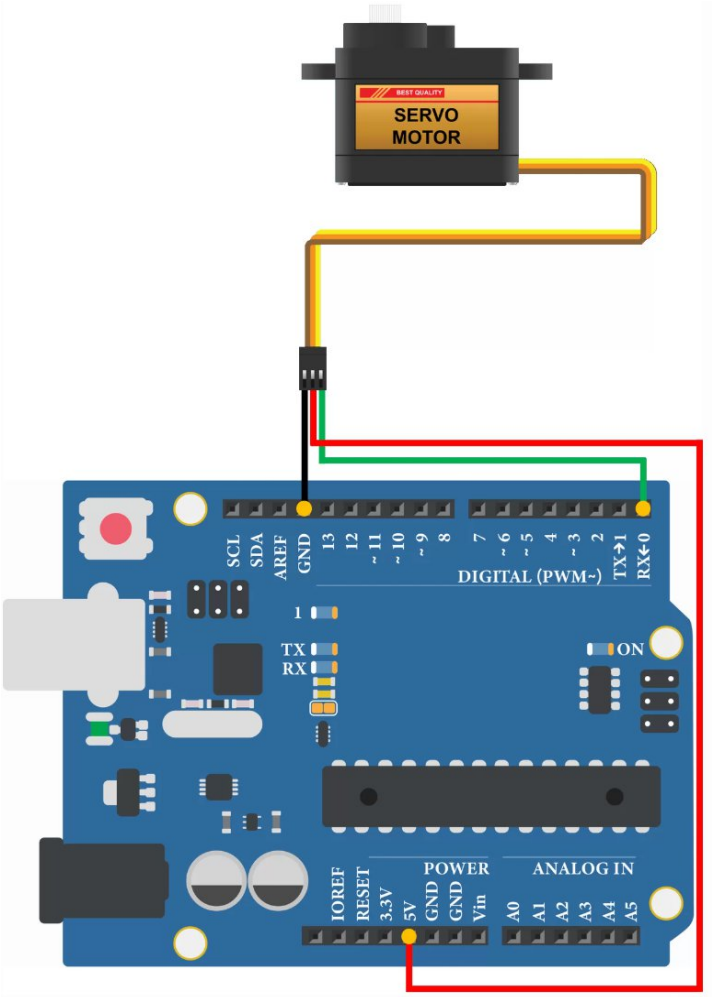

Рисунок 3.2 - Підключення сервоприводу до плати

Сервопривід керується за допомогою довжини сигналу, через що у нас немає необхідності в приєднанні додаткових модулів чи перетворювачів. Для живлення сервоприводів потрібно 5В, яке можна реалізувати через живлення плати, але в такому випадку доведеться зробити зміни в програму керування, аби переконатися, що система не спалить плату перенавантаживши її, для цього у ситуації якщо система не має зовнішнього живлення слід прописати в алгоритмі роботи перевірку та використовувати лише один сервопривід за раз,

що в свою чергу вплине на час роботи пристрою, але це не буде критичним та дозволить спростити електроніку за рахунок ускладнення коду.

Якщо буде прийнято рішення не використовувати ускладнення коду та використовувати сервоприводи одночасно розроблена система повинна живити сервоприводи від зовнішнього джерела живлення, оскільки Arduino не в змозі впоратися з струмом, який будуть споживати всі сервоприводи при роботі одночасно.

Джерело живлення повинно бути здатне витримувати струм не менше 2А для стабільного електроживленя.

У випадку якщо буде вирішено використовувати зовнішнє додаткове живлення для сервоприводів, можливо використовувати акумулятори або групу батарейок розраховані на напругу в 7В, але за заявами виробників та документацією Arduino MEGA здатна працювати при напрузі до 20В. Найпростіший спосіб виконати подібне підключення - використання групи батарейок + з яких подається на червоний кабель усіх сервоприводів та на VIN плати, відповідно земля подається на всі коричневі та на GND плати.

Для виконання керування обрано для використання стандартний Bluetooth модуль який підключається також напряму до плати та може бути спокійно заживлений від плати.

Підключення Bluetooth модулів до плати зображено на рисунку 3.3. Можливо використання модулю з чотирма пінами та з шістьма пінами, для наших вимог різниця невелика. Слід під’єднати чотири з шести або чотири з чотирьох пінів відповідно, в першу чергу при збірці слід з’єднати землю обох приладів, після чого приєднується живлення VCC Bluetooth модулю до 5V Arduino MEGA, далі необхідно з'єднати TX та RX модулю з пінами які будуть використовуватися в коді, на рисунку 3.3 зображено під’єднання до виходів 3 та 4.

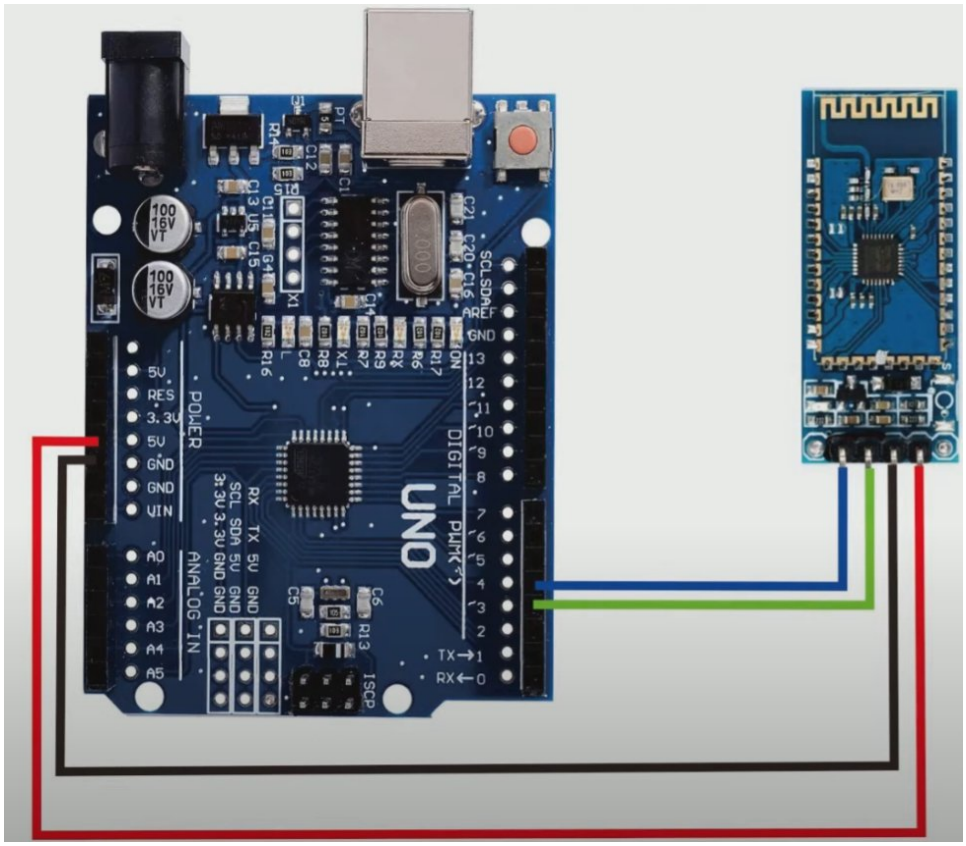

Рисунок 3.3 - Підключення Bluetooth модулю

Таким чином з’єднавши усі елементи отримано схему підключення робота маніпулятора (рисунок 3.4) з використанням шести сервоприводів, зовнішнього живлення та Bluetooth модулю для виконання поставлених задач.

Як було зумовлено вимогами до розроблюваного завдання та аналізом елементної бази алгоритм в першу чергу після отримання початкових змінних буде проводити перевіряти який тип живлення прописано для плати. У випадку якщо використовується зовнішнє живлення для запуску сервоприводів система працює як зазвичай, якщо використовується живлення сервоприводів від плати, це означає що вводиться обмеження та системі забороняється використовувати більше одного сервоприводу за раз шляхом використання змінної VCC_rule = 1, яка буде викликатися при використання зміни положення - керування роботом маніпулятором.

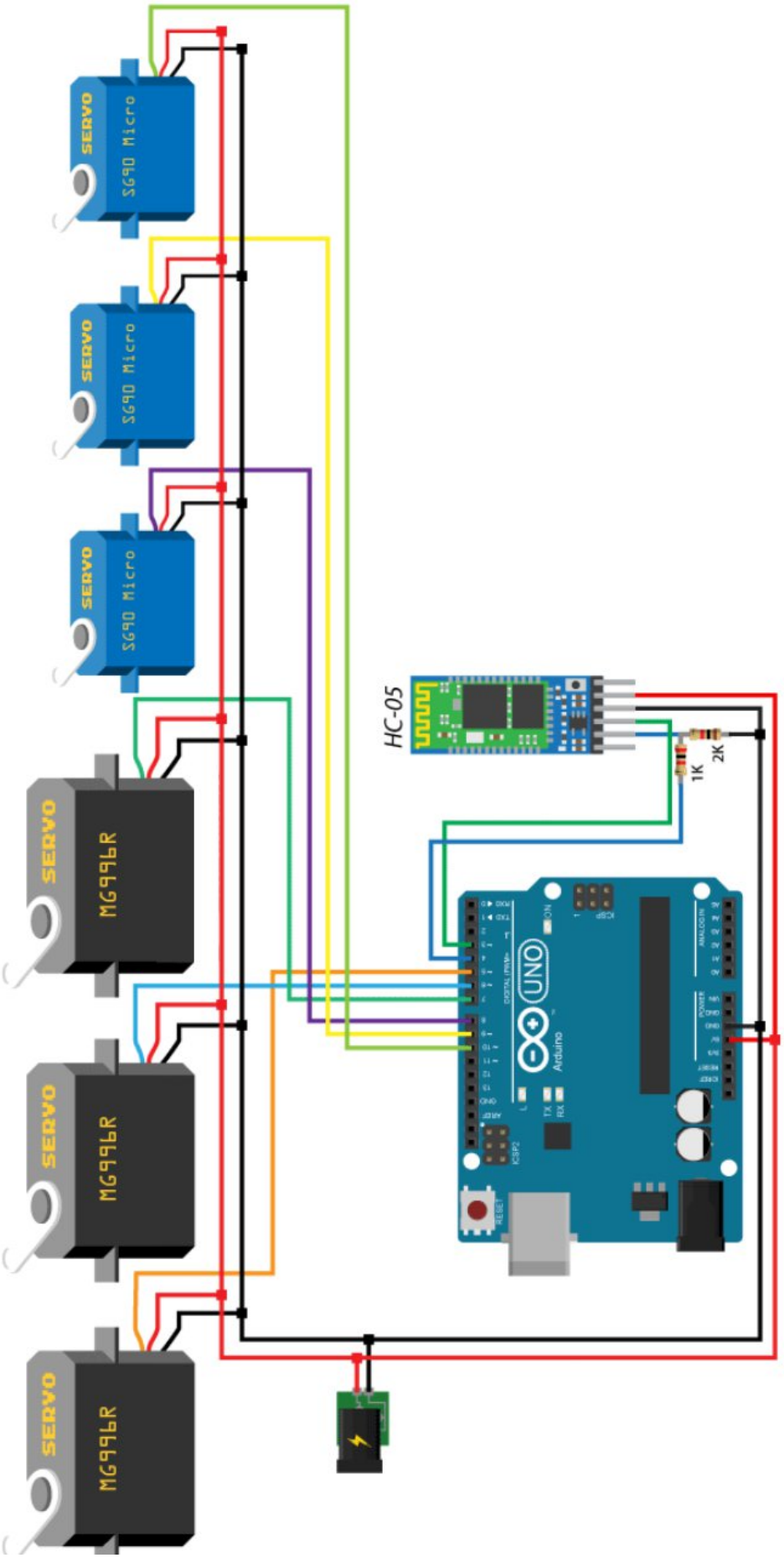

Рисунок 3.4 - Підключення робота маніпулятора

3.2 Розробка алгоритму керування системою зарядки дронів

Наступним кроком виконується перевірка типу контролю, якщо змінна Control встановлено на 1, це свідчить про використання ручного контролю з додатку, через що система очікує на команди від блютуз модулю для визначення наступних дій.

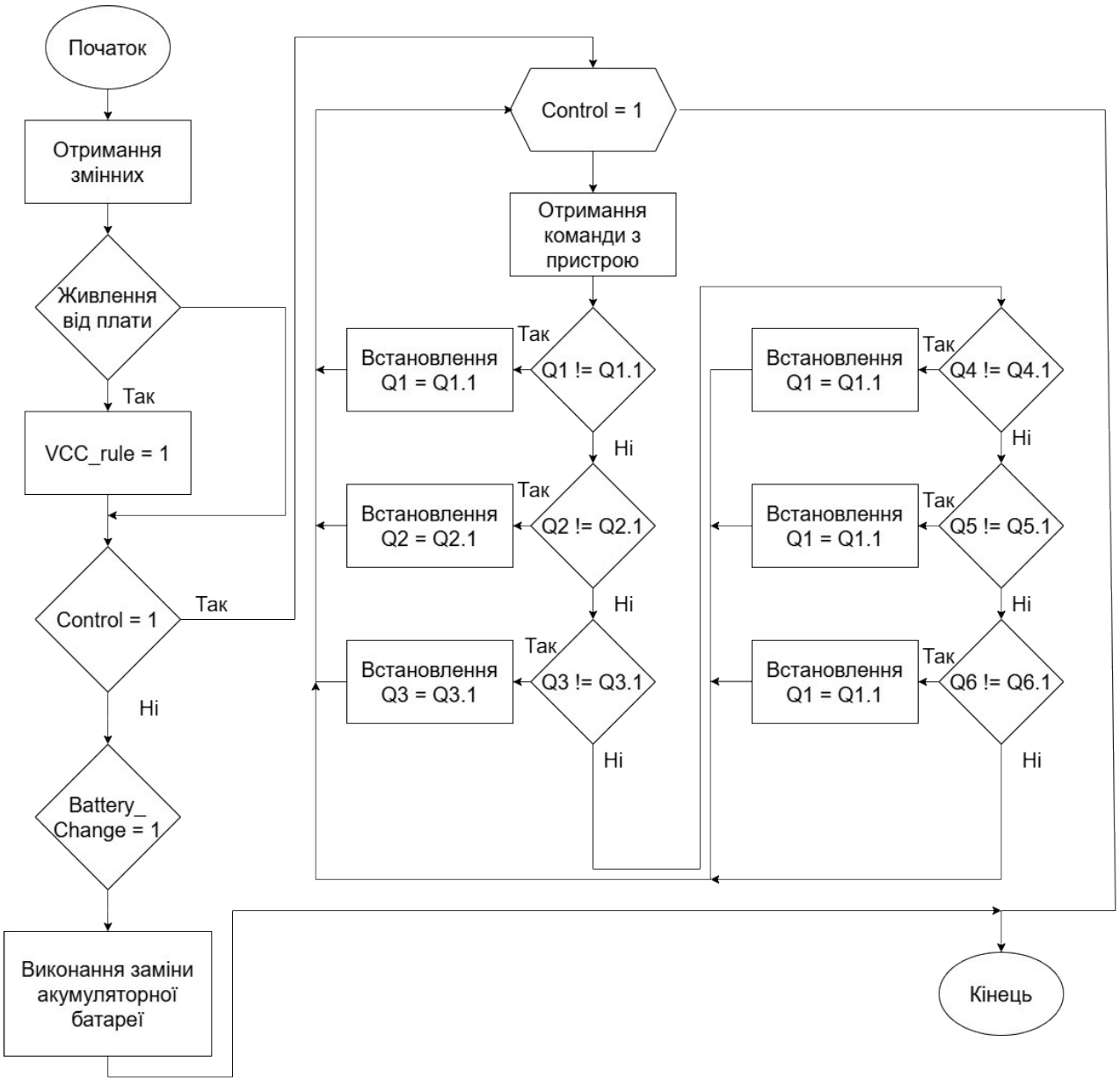

Рисунок 3.5 - Блок схема алгоритму керування програмою

У випадку якщо використовується автоматичне керування система переходить в стан перевірки чи потрібно виконати заміну акумулятору та очікуємо на з’явлення сигналу від дрону або користувача про виконання заміни.

Якщо Control = 1, переходимо до циклу while, вводячи постійну перевірку зміни сигналу для кожної змінної Q, які відповідають за положення сервоприводів. Якщо зчитується що задане пам’яттю та дійсне положення Q1 не дорівнює змінній Q1.1 яка отримується з програми керування на пристрої приєднаному через безпровідну мережу, система передає на сервопривід положення Q1 == Q1.1, що переводить сервопривід в нове задане положення.

Подібна перевірка виконується для кожного сервоприводу які пронумеровані з основи пристрою Q1, Q2, Q3, Q4, Q5, Q6 відповідно, при потребі виконується регулювання кожного з них послідовно для уникнення можливих помилок та зіткнень.

При завершенні виставлення положення програма входить в цикл та перевіряє дані додатку до моменту перемикання у стан автоматичної роботи.

Частину алгоритму програми відповідальну за визначення цих параметрів та перехід до різних типів контролю зображено на рисунку 3.5.

Наступним важливим етапом є встановлення ключових точок у роботі автоматичного виконання поставлених задач, таких як:

- стартова позиція (0);
- виставлення над закритою кришкою дрону (1);
- захоплення кришки дрону (2);
- підняття кришки дрону та відпуск кришки (3);
- виставлення над акумулятором у корпусі дрону (4);
- захоплення/вивільнення акумулятору в слот дрону (5, 5.1 відповідно);
- виставлення над зарядним слотом 1 (6.1);
- виставлення над зарядним слотом 2 (6.2);

- виставлення над зарядним слотом 3 (6.3);
- вивільнення/захоплення акумулятору в слот зарядки (7, 7.1 відповідно);
- виставлення над відкритою кришкою дрону (8);
- захоплення відкритої кришки дрону (9);
- зачинення кришки дрону (10);

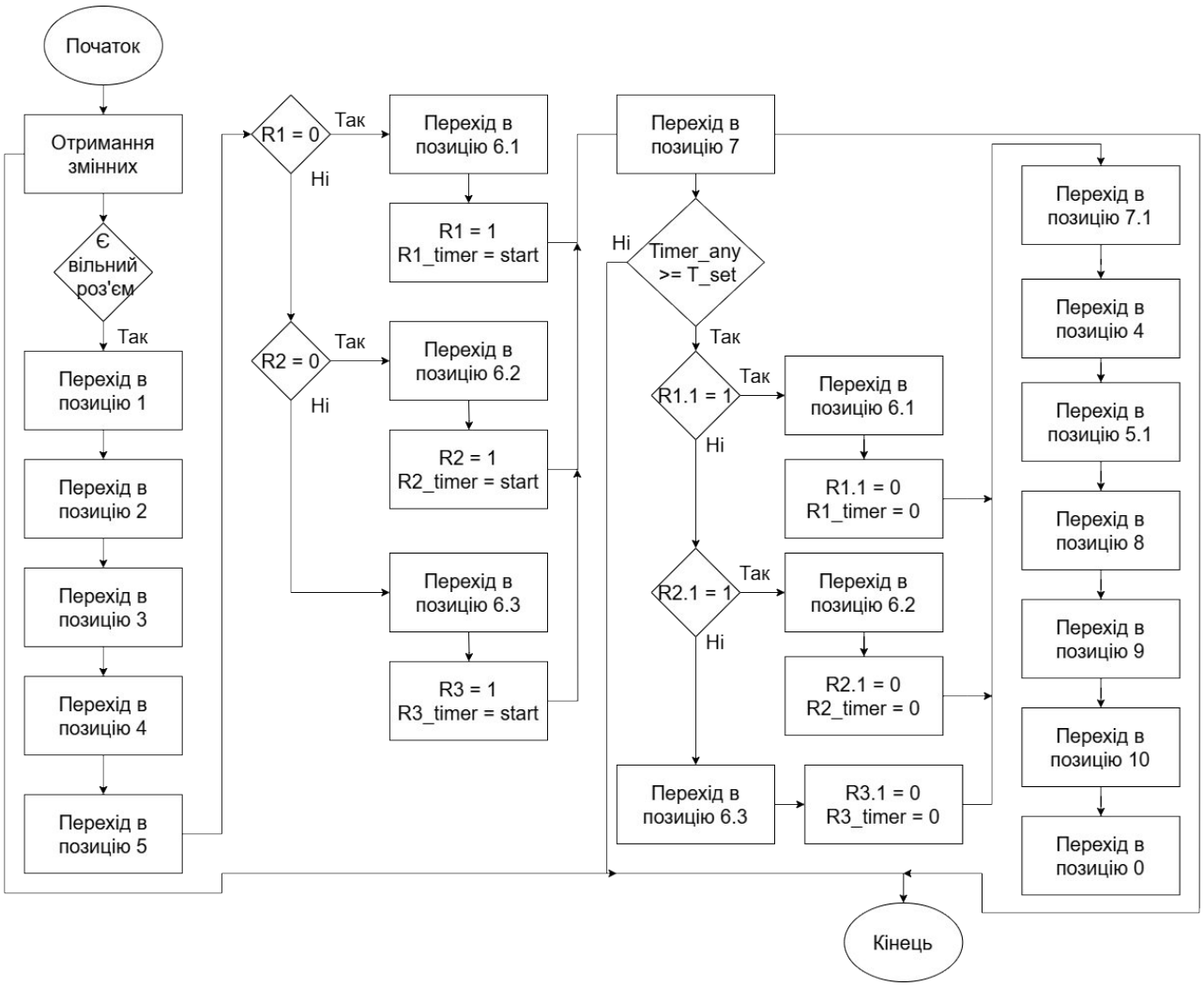

Рисунок 3.6 - Блок схема алгоритму керування роботом маніпулятором

Блок схему роботи роботом маніпулятором зображено на рисунок 3.6. Умовне позиціонування сервоприводів відображено в таблиці 3.1. Якщо умикаєтеся функція заміни акумулятору з попереднього алгоритму в першу

чергу відбувається перевірка змінних R1 R2 R3, якщо вони всі стоять на 1 це означає що всі зарядні слоти зайняти і система не можемо виконати заміну акумуляторної батареї для перезарядки дрону.

На даній стадії наша система передбачає використання двох запасних батарей, як результат у нас завжди буде один вільний слот, але можлива модифікація з використанням підставки на яку буде встановлено розряджену акумуляторну батарею з дрону, після чого один з роз'ємів зарядного пристрою буде звільнено а заряджену батарею поміщено в слот дрону, після чого батарею розряджену буде переміщено в зарядний слот який звільнився.

У випадку наявності вільного роз'єму система виконує наступну послідовність дій:

1. Виставлення робо-клешні над закритою кришкою дрону в відкритій позиції клешні (P1);
2. Закриття робо-клешні для захоплення кришки дрону шляхом зміни положення сервоприводів Q6 та Q5(P2);
3. Підняття кришки дрону позиції замкнутої клешні та розжаття клешні в кінцевій позиції для відпуску кришки (P3);
4. Виставлення над акумулятором у корпусі дрону на висоті 5 см у відкритому стані клешні (P4);
5. Спуск на 5 см для захоплення акумулятору в слот дрону (P5);
6. Визначення вільного слота зарядки:
    a. Виставлення над зарядним слотом 1 стан клешні закритий, перезапис змінної відповідної за зайнятість слоту R1.1=1, видалення таймеру для фіксації часу зарядки R1_timer (P6.1);
    b. Виставлення над зарядним слотом 2 стан клешні закритий, перезапис змінної відповідної за зайнятість слоту R2.1=1, видалення таймеру для фіксації часу зарядки R2_timer (P6.2);

c. Виставлення над зарядним слотом 3 стан клешні закритий, перезапис змінної відповідної за зайнятість слоту R3.1=1, видалення таймеру для фіксації часу зарядки R3_timer (P6.3);

7. Вивільнення акумулятору в слот зарядки шляхом розведення клешні (P7);

8. Перевірка параметру таймерів для визначення зарядженого акумулятору в слоті зарядки:

a. Виставлення над зарядним слотом 1 стан клешні закритий, перезапис змінної відповідної за зайнятість слоту R1=1, запуск таймеру для фіксації часу зарядки R1_timer (P6.1);

b. Виставлення над зарядним слотом 2 стан клешні закритий, перезапис змінної відповідної за зайнятість слоту R2=1, запуск таймеру для фіксації часу зарядки R2_timer (P6.2);

c. Виставлення над зарядним слотом 3 стан клешні закритий, перезапис змінної відповідної за зайнятість слоту R3=1, запуск таймеру для фіксації часу зарядки R3_timer (P6.3);

9. Захоплення акумулятору в слоті зарядки шляхом зведення клешні (P7);

10. Виставлення над акумулятором у корпусі дрону на висоті 5 см у відкритому стані клешні (P4);

11. Спуск на 5 см для вивільнення акумулятору в слот дрону (P5.1);

12. Виставлення над відкритою кришкою дрону в відкритому стані(P8);

13. Захоплення відкритої кришки дрону (P9);

14. Зачинення кришки дрону (P10);

15. Повернення в стартову позицію.

Задані в таблиці параметри положень сервоприводу розраховані при встановленні сервоприводів паралельно напрямку корпусу елементу оберту

роботу маніпулятору та встановленні заданої позиції за 0, через що при використанні та реалізації подібного проекту слід кілька разів перевірити та переконатися що сервоприводи якісні та правильно пронумеровані, оскільки бувають випадки коли сервопривід в стартовій позиції розташовано не на нульовій позиції, що в свою чергу призводить до неточності позиції.

Розташування робота маніпулятора передбачає вільний хід для елементів, оскільки немає необхідності встановлювати усі елементи ідеально, адже у всіх позиціях кришка дрону здатна зачинитися за рахунок енерції як і акумуляторна батарея. Точки ключових позицій задаются програмно та можуть бути відредаговані в любий момент.

Таблиця 3.1 - Умовні параметри положення сервоприводів

	Сервопривід					
	Q1	Q2	Q3	Q4	Q5	Q6
P0	0	0	0	0	0	0
P 1	0	90	0	0	20	90
P 2	0	90	45	90	0	90
P 3	0	100	50	90	90	180
P 4	0	88	48	45	0	90
P 5	0	87	48	45	0	90
P 5.1	0	87	48	45	90	180
P 6.1	180	90	90	45	90	180
P 6.2	180	90	90	45	90	180

Кінець таблиці 3.1

P 6.3	180	90	90	45	90	180
P 7	180	90	90	90	180	90
P 7.1	180	90	90	90	180	180
P 8	0	90	45	90	180	90
P 9	0	100	50	90	180	180
P 10	0	90	45	90	180	90

3.3 Розробка програми керування автоматизованої система зарядки дронів

Код написаний у додатку arduino-ide на мові Arduino programming language, спрощеній C++ з використанням бібліотеки ArduinoAPI. Хоча ця мова програмування називається Arduino programming language, насправді це C++ компілятор (avr-g++) з вбудованим додаванням main() і обгортки [39].

Спочатку потрібно під’єднати бібліотеку SoftwareSerial для послідовного зв'язку модуля Bluetooth, а також бібліотеку сервоприводів Servo.h. Ці бібліотеки входять до складу Arduino IDE, тому їх не потрібно встановлювати а просто увімкнути в коді.

Наступним кроком задаються входи, тим самим визначаємо в пам’яті як змінні усі елементи системи, шість сервоприводів, Bluetooth-модуль і деякі змінні для зберігання поточного і попереднього положення сервоприводів, а також масиви для зберігання положень для автоматичного режиму рисунок 3.7.

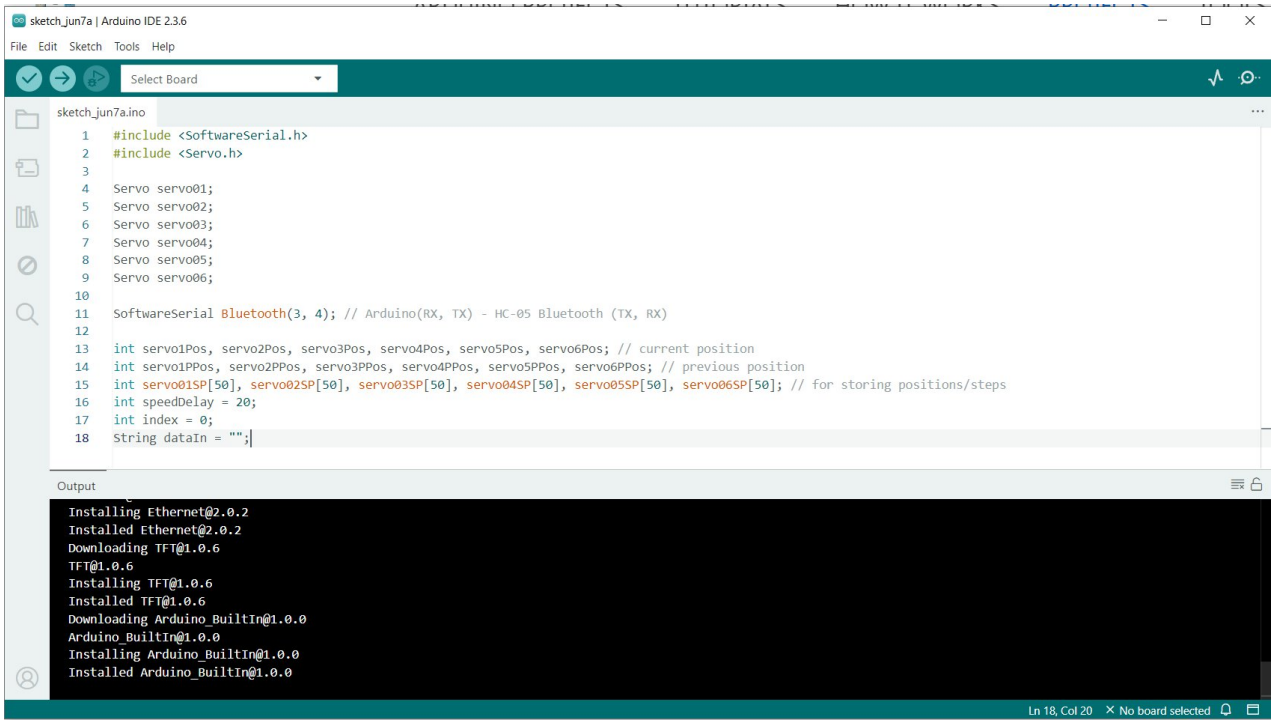

Рисунок 3.7 - Код під'єднання бібліотек та ініціалізації змінних в Arduino IDE

У Void setup() потрібно ініціалізувати сервоприводи і Bluetooth-модуль, а також перемістити руку робота в початкове положення. Виконується це за допомогою функції write(), яку алгоритм використовує для переміщення сервоприводу у положення від 0 до 180 градусів, приписуючи кут у змінну servo1PPos = 90 та викликаючи servo01.write(servo1PPos) (рисунок 3.8) [40].

Після цього в розділі циклів за допомогою функції Bluetooth.available() алгоритм постійно робить перевірку чи є вхідні дані від контролюючого пристрою. Якщо він фіксується, то за допомогою функції readString() зчитується дані у вигляді рядка і зберігаються їх у змінній dataIn. Залежно від отриманих даних алгоритм дає команду руці робота, що робити, if (Bluetooth.available() > 0) {dataIn = Bluetooth.readString().

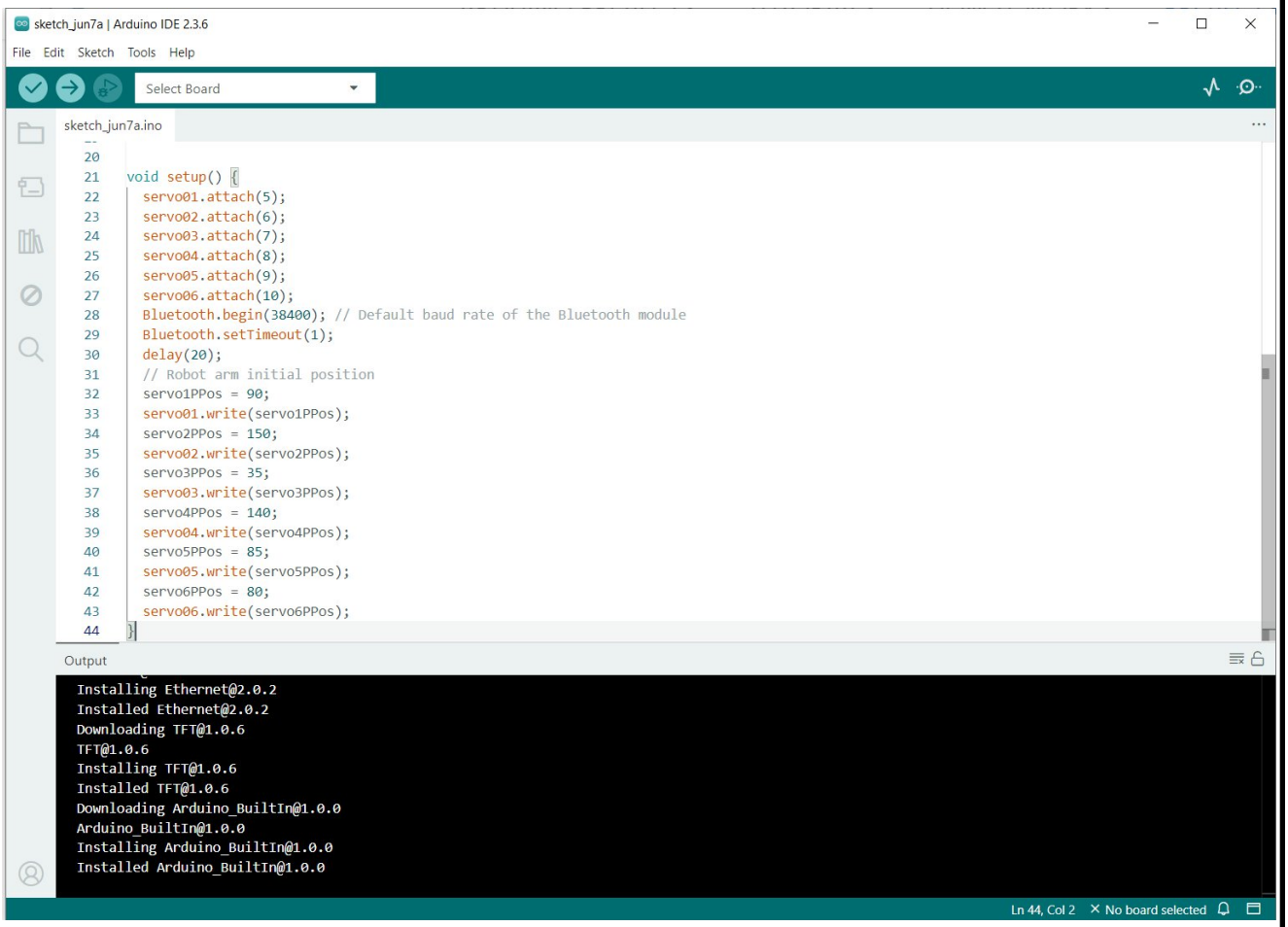

Рисунок 3.8 - Код Void setup() в Arduino IDE

Додаток було розроблено в MIT App Inventor online, оскільки ця мова та додаток дозволяють легко за допомогою блоків розробити функції для економії часу. MIT App Inventor - це безкоштовне онлайн-середовище для створення застосунків під Android і IOS(частково), створене Массачусетським технологічним інститутом. Платформа створена для розширення доступності програмування для всіх, та дозволяє розробляти програми без досвіду у розробці програмного забезпечення та без витрат часу на розробку бекенду, зображення, інтерфейс MIT App Inventor online для реалізації додатку зображено на рисунку 3.9 [41].

Інтерфейс і логіка App Inventor складається з двох основних частин:

- Designer - візуальний редактор інтерфейсу, у ньому можна перетягувати та розмістити кнопки, поля введення, зображення, елементи навігації та безліч інших елементів;
- Blocks - середовище програмування, використовує блокову мову за допомогою якої користувачі створюють логіку застосунку, поєднуючи блоки, які відповідають за дії, умови, цикли, події тощо.

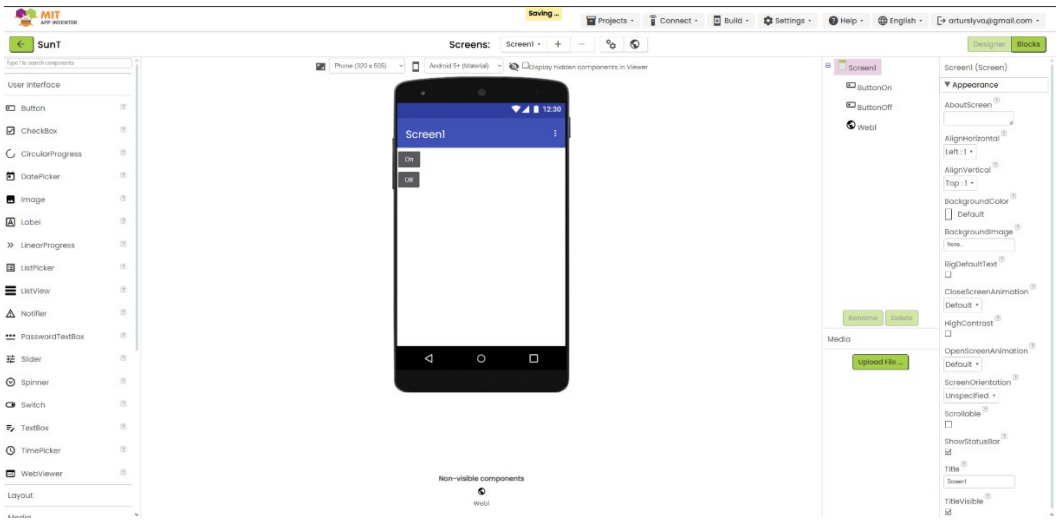

Рисунок 3.9 - Інтерфейс додатку в MIT App Inventor online

Цей підхід дозволяє зосередитися на логіці керування без необхідності писати код вручну. Він ідеально підходить для новачків, шкіл, університетів або швидкого прототипування.

Головними обмеженнями подібного додатку є:

- не вся функціональність доступна для iOS;
- не підходить для складних застосунків із високою продуктивністю що може стати проблемою при передачі інформації з багатьох повзунків чи при оновленні даних повзунків при автоматичні роботі програми;
- робота з дизайном на мінімальному рівні;

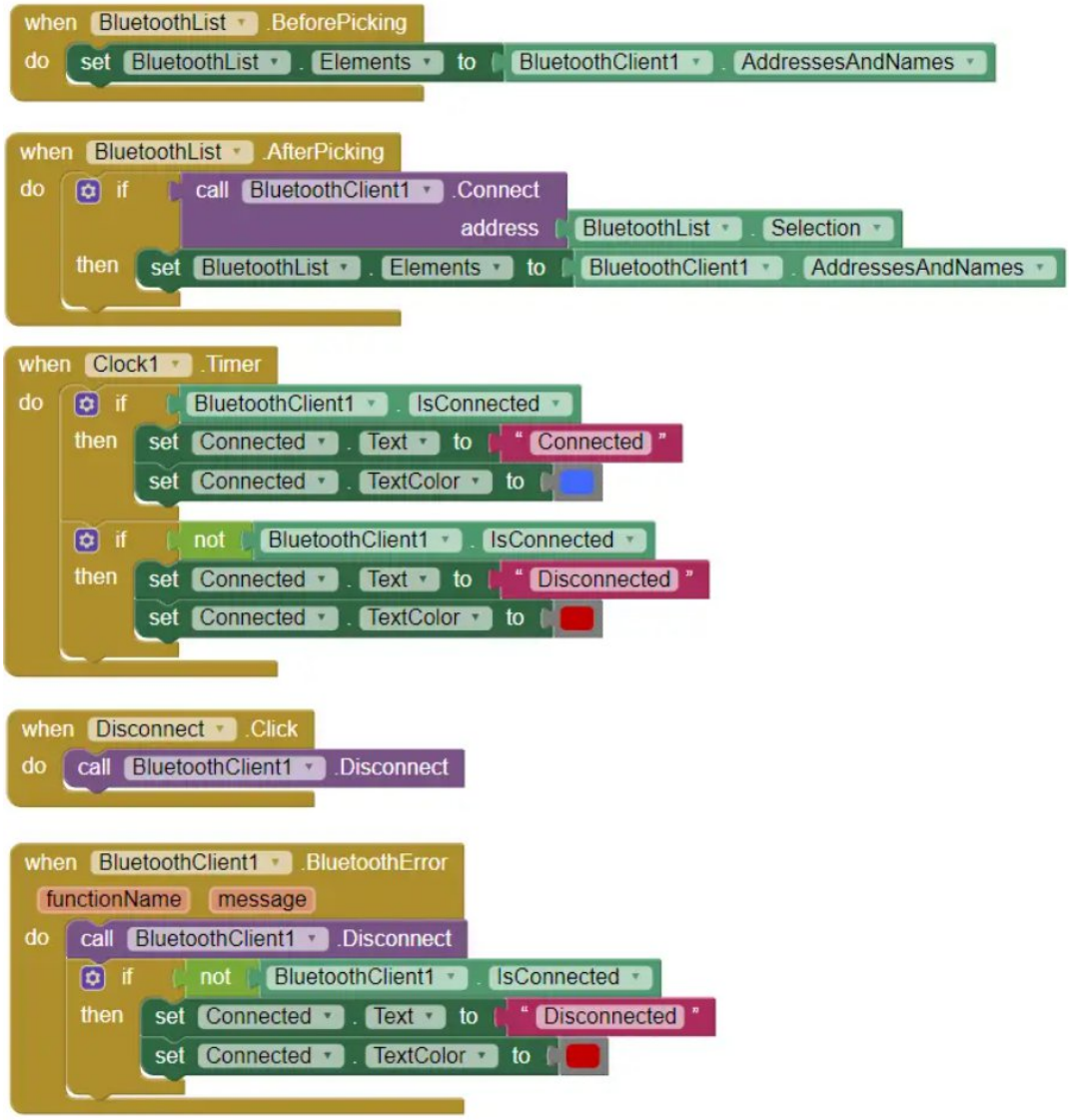

Рисунок 3.10 - Код підключення Bluetooth в MIT App Inventor online

Для візуалізації пристрою було використано готовий приклад для керування роботом маніпулятором з використанням наших параметрів (рисунок 3.10). Таким чином система зможео приєднати Bluetooth в MIT App Inventor online за допомогою блоків на рисунку 3.11.

Наступним кроком є налаштування блоків повзунків для керування положенням сервоприводу та блоки кнопок для вибору типу роботи та переходу на зазначені позиції (рисунок 3.12).

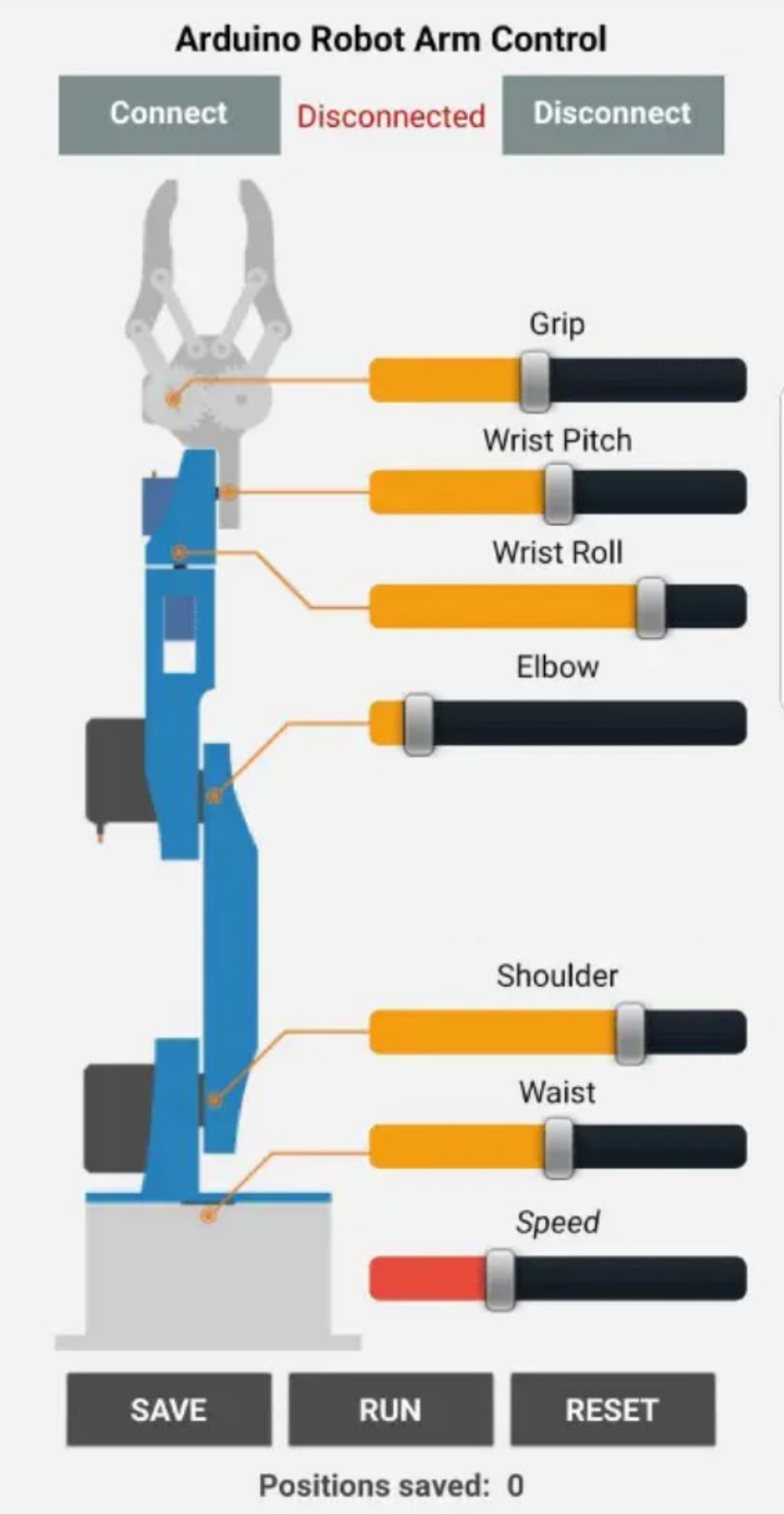

Рисунок 3.11 - Фронтенд додатку в MIT App Inventor online

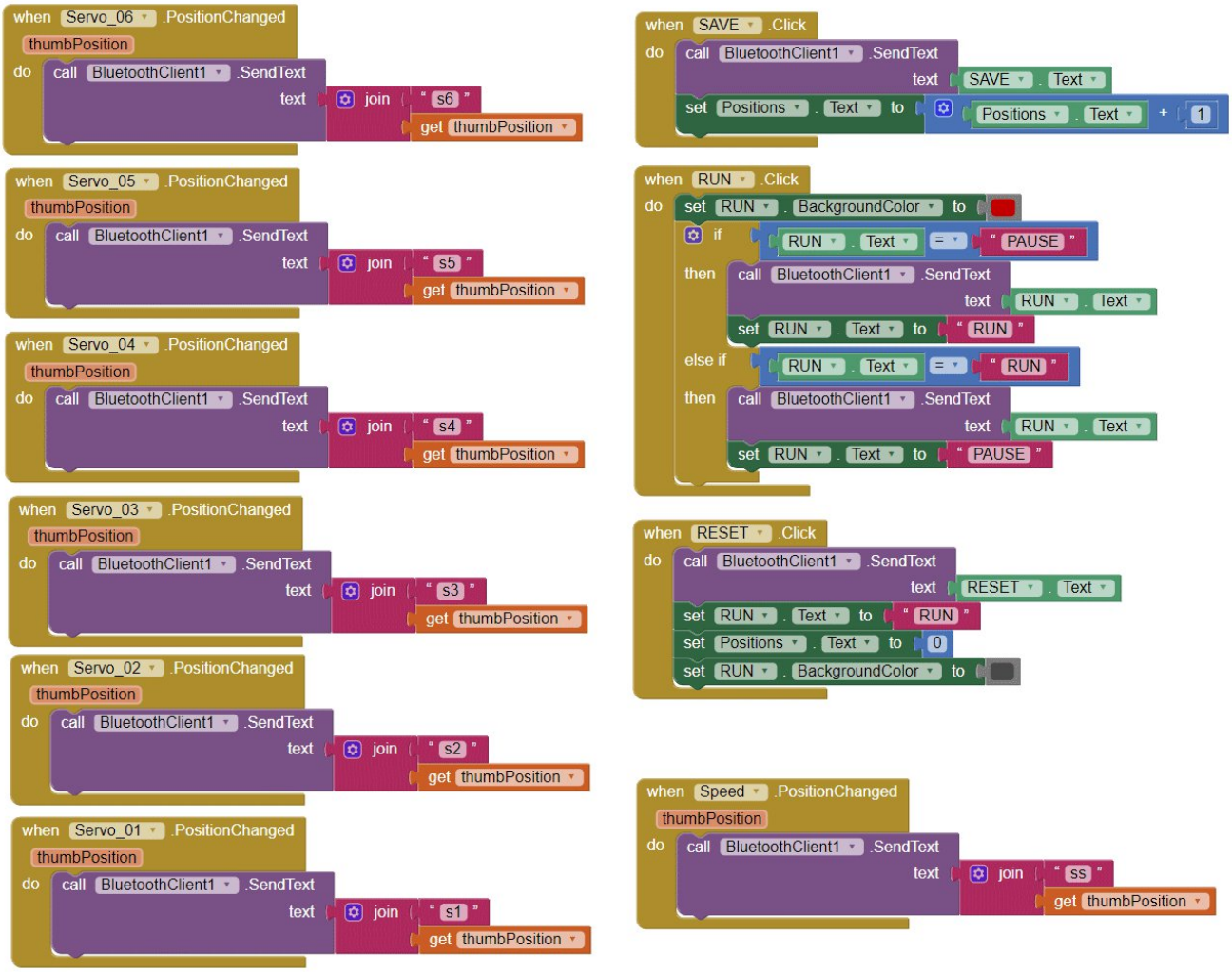

Рисунок 3.12 - Код керування повзунками в MIT App Inventor online

Якщо в додатку виконується змінна положення повзунка, за допомогою функції Bluetooth .SendText алгоритм надсилає текст на Arduino. Цей текст складається з префікса, який вказує на те, який повзунок було змінено, а також поточне значення повзунка.

Код за кожною з кнопок прописується так само в Arduino IDE, якщо користувач натисне кнопку RUN, алгоритм викликає користувацьку функцію runservo(), яка виконує збережені кроки. При запуску цієї функції алгоритм виконуємо збережені кроки знову і знову, до моменту натискання кнопку RESET. За допомогою циклу FOR алгоритм перебирає всі позиції, що

зберігаються в масивах, і одночасно перевіряємо, чи є у збережених параметрах якісь вхідні дані зі смартфона. Якщо користувач змінить положення повзунка, алгоритм буде використовувати це значення для зміни часу затримки між кожною ітерацією в циклах FOR нижче, які керують швидкістю сервомоторів.

Подібним чином, як було описану раніше, за допомогою операторів IF і циклів FOR та WHILE алгоритм відповідає за переміщення сервоприводів у наступну позицію.

Якщо користувач натисне на кнопку RESET, алгоритм виконає очищення всіх данних з масивів до нуля, а також встановимо index = 0, для зміни параметрів руки робота на нові рухи.

Таким чином реалізовано просте керування пристрою автоматизованої зарядки дронів, виконуючи керування робота маніпулятора за допомогою коду написаного на мові Arduino.

3.4 Висновки до третього розділу

У третьому розділі було детально розглянуто процес вибору, підключення та керування сервоприводами та іншими ключовими елементами автоматизованої системи.

Було обґрунтовано спосіб підключення сервоприводів MG996 та SG90 а також описано схему їхнього підключення до плати Arduino MEGA з урахуванням варіантів живлення як від плати, так і від зовнішнього джерела.

Проведено аналіз впливу типу живлення на можливість одночасної роботи декількох сервоприводів та на вплив збільшення навантаження на плату Arduino MEGA, а також наслідки перевищення допустимих параметрів навантаження.

Особливу увагу приділено реалізації Bluetooth-зв’язку для ручного керування маніпулятором, а також розробці алгоритму автоматичної заміни

акумуляторів. Алгоритм детально структуровано на основі ключових позицій маніпулятора та умов виконання завдань, що забезпечує точне та безпечне функціонування системи.

Було розроблено візуальну частину додатку за допомогою MIT App Inventor online для покращення якості ручного керування за рахунок візуалізації стану та розташування сервоприводів у системі автоматизованої зарядки дронів.

Реалізована схема підключення та логіка роботи повністю відповідає вимогам до автономної зарядної станції для дронів. Вона є гнучкою, масштабованою та забезпечує як автоматичний, так і ручний контроль з урахуванням обмежень апаратної частини.

ВИСНОВКИ

В процесі розробки автоматизованої зарядної станції для дронів у якості керуючого пристрою нами було обрано мікроконтролер Arduino Mega, що забезпечує керування виконавчими елементами та зв'язок з користувачем через Bluetooth.

У першому розділі було проведено огляд існуючих підходів до перезаряджання дронів. Розглянуто переваги та недоліки контактного заряджання, індукційного заряджання, використання сонячної енергії та повної автоматизованої заміни акумуляторів. Найперспективнішим підходом для систем які повинні бути в експлуатації якомога частіше визнано автоматизовану заміну акумуляторних батарей.

У другому розділі обґрунтовано вибір ключових апаратних компонентів системи. Для реалізації рухів маніпулятора обрано сервоприводи MG996R та SG90, які забезпечують необхідну потужність і точність. Також визначено типи конекторів, що використовуються для підключення акумулятора, з урахуванням механічного навантаження та потреби в балансуванні елементів.

Запропоновано конструктивні рішення для фіксації акумулятора, які враховують можливість 3D-друку та модульного підходу.

У третьому розділі розроблено алгоритм автоматичної заміни акумулятора, реалізовано схеми підключення виконавчих елементів до плати Arduino Mega та Bluetooth-модуля.

Система підтримує як повністю автономний режим, так і ручне керування для налаштування. Розроблена система є гнучкою та придатною до використання в різних сценаріях експлуатації.

Завдяки використанню позиціонування користувачі здатні модифікувати систему для сумісності з конкретними типами дронів, що дозволяє значно зменшити час простою та підвищити загальну ефективність роботи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Чим відрізняються дрони, БПЛА, безпілотники, квадрокоптери та радіокеровані моделі. *Прямі дистрибуції*. URL: https://distributions.com.ua/ua/reviews/v-drone (дата звернення: 08.06.2025).
2. Royal Mail починає використовувати безпілотні літальні апарати. *https://logist.fm*. URL: https://logist.fm/news/royal-mail-pochinaie-vikoristovuvati-bezpilotni-litalni-aparati (дата звернення: 08.06.2025).
3. Ukrinform. Британська пошта використовує дрони для доставки між островами. *Укрінформ - актуальні новини України та світу*. URL: https://www.ukrinform.ua/rubric-technology/3743597-britanska-posta-vikoristovue-droni-dla-dostavki-miz-ostrovami.html (дата звернення: 08.06.2025).
4. Нюанси застосування дронів у господарствах: підсумки сезону | Пропозиція – Головний журнал з питань агробізнесу. *Пропозиція – Головний журнал з питань агробізнесу*. URL: https://propozitsiya.com/articles/tekhnika-ta-obladnannya-inshe/nyuansy-zastosuvannya-droniv-u-hospodarstvakh-pidsumky (дата звернення: 08.06.2025).
5. Як дрони допомагають аграріям в 2025. *Агроэксперт-Трейд*. URL: https://agroexp.com.ua/uk/kak-drony-pomogayut-agrariyam (дата звернення: 08.06.2025).
6. Проф., д-р Петро Когут. Дрони і Супутники: Особливості Використання в Агробізнесі. *EOS Data Analytics*. URL: https://eos.com/uk/blog/drony-ta-suputnyky/ (дата звернення: 08.06.2025).
7. BEZPEKA.CLUB. Переваги та недоліки застосування дронів для охорони периметра. *bezpeka.club*. URL: https://bezpeka.club/perevagy-ta-nedoliky-zastosuvannya-droniv-dlya-ohorony-perymetra (дата звернення: 08.06.2025).

8. Liang Y., Mouli G. R. C., Bauer P. Charging Technology for Electric Aircraft: State of the Art, Trends, and Challenges. *IEEE Transactions on Transportation Electrification*. 2023. С. 1. URL: https://doi.org/10.1109/tte.2023.3333536 (дата звернення: 08.06.2025).

9. Product Focus: One Kilowatt Wireless Charging for Mobile Robots. *techbriefs*. URL: https://www.techbriefs.com/component/content/article/50681-product-focus-one-kilowatt-wireless-charging-for-mobile-robots (дата звернення: 08.06.2025).

10. Defence & Security drone - XSun. *XSun*. URL: https://xsun-fr.com/applications/defence-security/ (дата звернення: 08.06.2025).

11. Innovative, High-Performance DEUTSCH DT-XT Connectors. *te.co*. URL: https://www.te.com/en/products/connectors/automotive-connectors/intersection/dt-xt-sealed-connector-system.html?tab=pgp-story (дата звернення: 08.06.2025).

12. Aerial Dockable Multirotor UAVs: Design, Control and Flight Time Extension through In-flight Battery Replacement. Y. Song та ін. *IEEE Access*. 2025. С. 1. URL: https://doi.org/10.1109/access.2025.3574452

13. Integrated Lithium-Ion Battery Packs + V-Twin EFI Engines for Drones, UAVs & Robots. unmannedsystemstechnology. URL: https://www.unmannedsystemstechnology.com/company/vanguard/ (дата звернення: 08.06.2025).

14. DJI Dock 3: Features, pricing and availability. *drone-parts-center*. URL: https://drone-parts-center.com/en/blog/dji-dock-3-features-pricing-and-availability/?srsltid=AfmBOopXDMoPRG3EV_0YxsG_1M8SFZBkk_L9gbKkIU9YiFRJQuWSa4R1 (дата звернення: 08.06.2025).

15. DJI's First Dock Adaptable for Vehicle Mounting DJI Dock 3 DJI Dock 3 Rise to Any Challenge. *enterprise.dji*. URL: https://enterprise.dji.com/dock-3 (дата звернення: 08.06.2025).

16. DJI Dock 3 Delivers “Drone in a Box” Enterprise Solution For 24/7 Remote Operations. *Dji.* URL: https://www.dji.com/global/media-center/announcements/dji-release-dock-3 (дата звернення: 08.06.2025).

17. DJI Dock 3: The revolutionary autonomous station for drone deployment. *Dronexperts*. URL: https://www.dronexperts.com/en/article/dji-dock-3-the-revolutionary-autonomous-station-for-drone-deployment/ (дата звернення: 08.06.2025).

18. What is Arduino Mega 2560? A Getting Started Guide. *RS-DESIGNSPARK*. 2021 URL: https://www.rs-online.com/designspark/what-is-arduino-mega-2560-a-getting-started-guide (дата звернення: 08.06.2025).

19. Arduino Mega 2560 R3. Arduinokit. 2025 URL: https://arduinokit.com.ua/ua/p1080109370-arduino-mega-2560.html?srsltid=AfmBOoqB--K39zWQ9qd79RtDt2lYt-_ZlnThpMdFhw3svfWsj9CpFJHmp (дата звернення: 08.06.2025).

20. ATMEGA2560 Datasheet, PDF. *ALLDATASHEET.COM - Electronic Parts Datasheet Search*. URL: https://www.alldatasheet.com/view.jsp?Searchword=Atmega2560%20datasheet&gad_source=1&gad_campaignid=160596484&gclid=Cj0KCQjw0qTCBhCmARIsAAj8C4b93nPVF8ZoxCZOK5eUCreo2s3vV8ggJI0SugD2m2skHnM6zA1n2nMaAn8JEALw_wcB (дата звернення: 08.06.2025).

21. Arduino Forum. *Arduino Forum*. URL: https://forum.arduino.cc (дата звернення: 09.06.2025).

22. Форум arduino.ua. *Arduino*. URL: https://forum.arduino.ua (дата звернення: 09.06.2025).

23. Wei C. Z., Mohd Zaman M. H., Ibrahim M. F. Visual Servo Algorithm Of Robot Arm For Pick And Place Application. Jurnal Kejuruteraan. 2024. Т. 36, № 3. С. 891–898. URL: https://doi.org/10.17576/jkukm-2024-36(3)-05

24. How to Use Servo Motors in Robot Arms. *CubeMars*. URL: https://www.cubemars.com/article-283How+to+Use+Servo+Motors+in+Robot+Arms.html (дата звернення: 09.06.2025).

25. Development of robotic arm control using Arduino controller. K. Chenchireddy. *IAES International Journal of Robotics and Automation (IJRA)*. 2024. Т. 13, № 3. С. 264.

26. Development of robotic arm control using Arduino controller. K. Chenchireddy et al. *IAES International Journal of Robotics and Automation (IJRA)*. 2024. Vol. 13, no. 3. P. 264.

27. Abdul Karim M. Z. B., Thamrin N. M. Servo Motor Controller using PID and Graphical User Interface on Raspberry Pi for Robotic Arm. *Journal of Physics: Conference Series*. 2022. Т. 2319, № 1. С. 012015.

28. A Detailed Guide to Robotic arm mechanism. *Collaborative robotic automation | Universal Robots Cobots*. URL: https://www.universal-robots.com/in/blog/robotic-arm-mechanism/ (дата звернення: 09.06.2025).

29. Characteristics based visual servo for 6DOF robot arm control / S. Tsuchida et al. *Cognitive Robotics*. 2021. Vol. 1. P. 76–82.

30. A Robo-hand prototype design gripping device within the framework of sustainable development. Y. Al-Sharo. *Indian journal of Engineering*. 2023. С. 13.

31. Safeea M., Neto P. Precise positioning of collaborative robotic manipulators using hand-guiding. The International Journal of Advanced Manufacturing Technology. 2022.

32. Yurova V. A., Velikoborets G., Vladyko A. Design and Implementation of an Anthropomorphic Robotic Arm Prosthesis. Technologies. 2022. Vol. 10, no. 5. P. 103.

33. Bronnikov A., Bendeberia M. Development of a manipulator kinematic model using Abb Robot Studio. *Innovative Technologies And Scientific Solutions For Industries*. 2024. № 4(30). С. 5–18.

34. Lindner T., Milecki A., Wyrwał D. Positioning of the Robotic Arm Using Different Reinforcement Learning Algorithms. *International Journal of Control, Automation and Systems*. 2021. T. 19, № 4. C. 1661–1676.

35. Functional implant positioning in total hip arthroplasty and the role of robotic-arm assistance. A. Fontalis et al. *International Orthopaedics*. 2022.

36. A low-cost digital twin-driven positioning error compensation method for industrial robotic arm. Z. Wu et al. *IEEE Sensors Journal*. 2022. P. 1. URL: https://doi.org/10.1109/jsen.2022.3213428

37. Digital twin-driven 3D position information mutuality and positioning error compensation for robotic armю. Z. Wu et al. *IEEE Sensors Journal*. 2023. P. 1.

38. A positioning error compensation method for multiple degrees of freedom robot arm based on the measured and target position error. Y. Tian та ін. *Advances in Mechanical Engineering*. 2022. T. 14, № 5. C. 168781322210900.

39. Download and install Arduino IDE. *support.arduino*. URL: https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE (дата звернення: 10.06.2025).

40. Arduino Documentation - Void. *arduino*. URL: https://docs.arduino.cc/language-reference/en/structure/sketch/setup/ (дата звернення: 10.06.2025).

41. MIT App Inventor Main. *MIT App Inventor*. URL: https://appinventor.mit.edu (дата звернення: 10.06.2025).

Додаток А
Спрощений код Arduino IDE для контролю робота маніпулятору

```
#include <SoftwareSerial.h>
#include <Servo.h>

Servo servo01;
Servo servo02;
Servo servo03;
Servo servo04;
Servo servo05;
Servo servo06;

SoftwareSerial Bluetooth(3, 4); // Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // current position
int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // previous position
int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50], servo06SP[50]; // for storing positions/steps
int speedDelay = 20;
int index = 0;
String dataIn = "";

void setup() {
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(38400); // Default baud rate of the Bluetooth module
  Bluetooth.setTimeout(1);
  delay(20);
  // Robot arm initial position
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 150;
  servo02.write(servo2PPos);
  servo3PPos = 35;
  servo03.write(servo3PPos);
```

```
  servo4PPos = 140;
  servo04.write(servo4PPos);
  servo5PPos = 85;
  servo05.write(servo5PPos);
  servo6PPos = 80;
  servo06.write(servo6PPos);
}

void loop() {
  // Check for incoming data
  if (Bluetooth.available() > 0) {
    dataIn = Bluetooth.readString();  // Read the data as string

    // If "Waist" slider has changed value - Move Servo 1 to position
    if (dataIn.startsWith("s1")) {
      String dataInS = dataIn.substring(2, dataIn.length()); // Extract only the number. E.g. from "s1120" to "120"
      servo1Pos = dataInS.toInt();  // Convert the string into integer
      // We use for loops so we can control the speed of the servo
      // If previous position is bigger then current position
      if (servo1PPos > servo1Pos) {
        for ( int j = servo1PPos; j >= servo1Pos; j--) {   // Run servo down
          servo01.write(j);
          delay(20);    // defines the speed at which the servo rotates
        }
      }
      // If previous position is smaller then current position
      if (servo1PPos < servo1Pos) {
        for ( int j = servo1PPos; j <= servo1Pos; j++) {   // Run servo up
          servo01.write(j);
          delay(20);
        }
      }
      servo1PPos = servo1Pos;   // set current position as previous position
    }

    // Move Servo 2
    if (dataIn.startsWith("s2")) {
      String dataInS = dataIn.substring(2, dataIn.length());
      servo2Pos = dataInS.toInt();
```

```
  if (servo2PPos > servo2Pos) {
    for ( int j = servo2PPos; j >= servo2Pos; j--) {
      servo02.write(j);
      delay(50);
    }
  }
  if (servo2PPos < servo2Pos) {
    for ( int j = servo2PPos; j <= servo2Pos; j++) {
      servo02.write(j);
      delay(50);
    }
  }
  servo2PPos = servo2Pos;
}
// Move Servo 3
if (dataIn.startsWith("s3")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo3Pos = dataInS.toInt();
  if (servo3PPos > servo3Pos) {
    for ( int j = servo3PPos; j >= servo3Pos; j--) {
      servo03.write(j);
      delay(30);
    }
  }
  if (servo3PPos < servo3Pos) {
    for ( int j = servo3PPos; j <= servo3Pos; j++) {
      servo03.write(j);
      delay(30);
    }
  }
  servo3PPos = servo3Pos;
}
// Move Servo 4
if (dataIn.startsWith("s4")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo4Pos = dataInS.toInt();
  if (servo4PPos > servo4Pos) {
    for ( int j = servo4PPos; j >= servo4Pos; j--) {
```

```
      servo04.write(j);
      delay(30);
    }
  }
  if (servo4PPos < servo4Pos) {
    for ( int j = servo4PPos; j <= servo4Pos; j++) {
      servo04.write(j);
      delay(30);
    }
  }
  servo4PPos = servo4Pos;
}
// Move Servo 5
if (dataIn.startsWith("s5")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo5Pos = dataInS.toInt();
  if (servo5PPos > servo5Pos) {
    for ( int j = servo5PPos; j >= servo5Pos; j--) {
      servo05.write(j);
      delay(30);
    }
  }
  if (servo5PPos < servo5Pos) {
    for ( int j = servo5PPos; j <= servo5Pos; j++) {
      servo05.write(j);
      delay(30);
    }
  }
  servo5PPos = servo5Pos;
}
// Move Servo 6
if (dataIn.startsWith("s6")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo6Pos = dataInS.toInt();
  if (servo6PPos > servo6Pos) {
    for ( int j = servo6PPos; j >= servo6Pos; j--) {
      servo06.write(j);
      delay(30);
    }
```

```
    }
    if (servo6PPos < servo6Pos) {
     for ( int j = servo6PPos; j <= servo6Pos; j++) {
      servo06.write(j);
      delay(30);
     }
    }
    servo6PPos = servo6Pos;
   }
   // If button "SAVE" is pressed
   if (dataIn.startsWith("SAVE")) {
    servo01SP[index] = servo1PPos;  // save position into the array
    servo02SP[index] = servo2PPos;
    servo03SP[index] = servo3PPos;
    servo04SP[index] = servo4PPos;
    servo05SP[index] = servo5PPos;
    servo06SP[index] = servo6PPos;
    index++;                    // Increase the array index
   }
   // If button "RUN" is pressed
   if (dataIn.startsWith("RUN")) {
    runservo();  // Automatic mode - run the saved steps
   }
   // If button "RESET" is pressed
   if ( dataIn == "RESET") {
    memset(servo01SP, 0, sizeof(servo01SP)); // Clear the array data to 0
    memset(servo02SP, 0, sizeof(servo02SP));
    memset(servo03SP, 0, sizeof(servo03SP));
    memset(servo04SP, 0, sizeof(servo04SP));
    memset(servo05SP, 0, sizeof(servo05SP));
    memset(servo06SP, 0, sizeof(servo06SP));
    index = 0;  // Index to 0
   }
  }
}

// Automatic mode custom function - run the saved steps
void runservo() {
  while (dataIn != "RESET") {   // Run the steps over and over again until "RESET" button is pressed
```

```
for (int i = 0; i <= index - 2; i++) {  // Run through all steps(index)
 if (Bluetooth.available() > 0) {      // Check for incomding data
  dataIn = Bluetooth.readString();
  if ( dataIn == "PAUSE") {           // If button "PAUSE" is pressed
   while (dataIn != "RUN") {          // Wait until "RUN" is pressed again
    if (Bluetooth.available() > 0) {
     dataIn = Bluetooth.readString();
     if ( dataIn == "RESET") {
      break;
     }
    }
   }
  }
  // If speed slider is changed
  if (dataIn.startsWith("ss")) {
   String dataInS = dataIn.substring(2, dataIn.length());
   speedDelay = dataInS.toInt(); // Change servo speed (delay time)
  }
 }
 // Servo 1
 if (servo01SP[i] == servo01SP[i + 1]) {
 }
 if (servo01SP[i] > servo01SP[i + 1]) {
  for ( int j = servo01SP[i]; j >= servo01SP[i + 1]; j--) {
   servo01.write(j);
   delay(speedDelay);
  }
 }
 if (servo01SP[i] < servo01SP[i + 1]) {
  for ( int j = servo01SP[i]; j <= servo01SP[i + 1]; j++) {
   servo01.write(j);
   delay(speedDelay);
  }
 }

 // Servo 2
 if (servo02SP[i] == servo02SP[i + 1]) {
 }
 if (servo02SP[i] > servo02SP[i + 1]) {
```

```
  for ( int j = servo02SP[i]; j >= servo02SP[i + 1]; j--) {
    servo02.write(j);
    delay(speedDelay);
  }
}
if (servo02SP[i] < servo02SP[i + 1]) {
  for ( int j = servo02SP[i]; j <= servo02SP[i + 1]; j++) {
    servo02.write(j);
    delay(speedDelay);
  }
}

// Servo 3
if (servo03SP[i] == servo03SP[i + 1]) {
}
if (servo03SP[i] > servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j >= servo03SP[i + 1]; j--) {
    servo03.write(j);
    delay(speedDelay);
  }
}
if (servo03SP[i] < servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j <= servo03SP[i + 1]; j++) {
    servo03.write(j);
    delay(speedDelay);
  }
}

// Servo 4
if (servo04SP[i] == servo04SP[i + 1]) {
}
if (servo04SP[i] > servo04SP[i + 1]) {
  for ( int j = servo04SP[i]; j >= servo04SP[i + 1]; j--) {
    servo04.write(j);
    delay(speedDelay);
  }
}
if (servo04SP[i] < servo04SP[i + 1]) {
  for ( int j = servo04SP[i]; j <= servo04SP[i + 1]; j++) {
```

```
    servo04.write(j);
    delay(speedDelay);
  }
}

// Servo 5
if (servo05SP[i] == servo05SP[i + 1]) {
}
if (servo05SP[i] > servo05SP[i + 1]) {
  for ( int j = servo05SP[i]; j >= servo05SP[i + 1]; j--) {
    servo05.write(j);
    delay(speedDelay);
  }
}
if (servo05SP[i] < servo05SP[i + 1]) {
  for ( int j = servo05SP[i]; j <= servo05SP[i + 1]; j++) {
    servo05.write(j);
    delay(speedDelay);
  }
}

// Servo 6
if (servo06SP[i] == servo06SP[i + 1]) {
}
if (servo06SP[i] > servo06SP[i + 1]) {
  for ( int j = servo06SP[i]; j >= servo06SP[i + 1]; j--) {
    servo06.write(j);
    delay(speedDelay);
  }
}
if (servo06SP[i] < servo06SP[i + 1]) {
  for ( int j = servo06SP[i]; j <= servo06SP[i + 1]; j++) {
    servo06.write(j);
    delay(speedDelay);
```

Додаток А

Спрощений код Arduino IDE для контролю робота маніпулятору

```
#include <SoftwareSerial.h>
#include <Servo.h>

Servo servo01;
Servo servo02;
Servo servo03;
Servo servo04;
Servo servo05;
Servo servo06;

SoftwareSerial Bluetooth(3, 4); // Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // current position
int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // previous position
int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50], servo06SP[50]; // for storing positions/steps
int speedDelay = 20;
int index = 0;
String dataIn = "";

void setup() {
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(38400); // Default baud rate of the Bluetooth module
  Bluetooth.setTimeout(1);
  delay(20);
  // Robot arm initial position
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 150;
  servo02.write(servo2PPos);
  servo3PPos = 35;
  servo03.write(servo3PPos);
```

```
  servo4PPos = 140;
  servo04.write(servo4PPos);
  servo5PPos = 85;
  servo05.write(servo5PPos);
  servo6PPos = 80;
  servo06.write(servo6PPos);
}

void loop() {
  // Check for incoming data
  if (Bluetooth.available() > 0) {
    dataIn = Bluetooth.readString();  // Read the data as string

    // If "Waist" slider has changed value - Move Servo 1 to position
    if (dataIn.startsWith("s1")) {
      String dataInS = dataIn.substring(2, dataIn.length()); // Extract only the number. E.g. from "s1120" to "120"
      servo1Pos = dataInS.toInt();  // Convert the string into integer
      // We use for loops so we can control the speed of the servo
      // If previous position is bigger then current position
      if (servo1PPos > servo1Pos) {
        for ( int j = servo1PPos; j >= servo1Pos; j--) {   // Run servo down
          servo01.write(j);
          delay(20);    // defines the speed at which the servo rotates
        }
      }
      // If previous position is smaller then current position
      if (servo1PPos < servo1Pos) {
        for ( int j = servo1PPos; j <= servo1Pos; j++) {   // Run servo up
          servo01.write(j);
          delay(20);
        }
      }
      servo1PPos = servo1Pos;   // set current position as previous position
    }

    // Move Servo 2
    if (dataIn.startsWith("s2")) {
      String dataInS = dataIn.substring(2, dataIn.length());
      servo2Pos = dataInS.toInt();
```

```
  if (servo2PPos > servo2Pos) {
    for ( int j = servo2PPos; j >= servo2Pos; j--) {
      servo02.write(j);
      delay(50);
    }
  }
  if (servo2PPos < servo2Pos) {
    for ( int j = servo2PPos; j <= servo2Pos; j++) {
      servo02.write(j);
      delay(50);
    }
  }
  servo2PPos = servo2Pos;
}
// Move Servo 3
if (dataIn.startsWith("s3")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo3Pos = dataInS.toInt();
  if (servo3PPos > servo3Pos) {
    for ( int j = servo3PPos; j >= servo3Pos; j--) {
      servo03.write(j);
      delay(30);
    }
  }
  if (servo3PPos < servo3Pos) {
    for ( int j = servo3PPos; j <= servo3Pos; j++) {
      servo03.write(j);
      delay(30);
    }
  }
  servo3PPos = servo3Pos;
}
// Move Servo 4
if (dataIn.startsWith("s4")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo4Pos = dataInS.toInt();
  if (servo4PPos > servo4Pos) {
    for ( int j = servo4PPos; j >= servo4Pos; j--) {
```

```
      servo04.write(j);
      delay(30);
    }
  }
  if (servo4PPos < servo4Pos) {
    for ( int j = servo4PPos; j <= servo4Pos; j++) {
      servo04.write(j);
      delay(30);
    }
  }
  servo4PPos = servo4Pos;
}
// Move Servo 5
if (dataIn.startsWith("s5")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo5Pos = dataInS.toInt();
  if (servo5PPos > servo5Pos) {
    for ( int j = servo5PPos; j >= servo5Pos; j--) {
      servo05.write(j);
      delay(30);
    }
  }
  if (servo5PPos < servo5Pos) {
    for ( int j = servo5PPos; j <= servo5Pos; j++) {
      servo05.write(j);
      delay(30);
    }
  }
  servo5PPos = servo5Pos;
}
// Move Servo 6
if (dataIn.startsWith("s6")) {
  String dataInS = dataIn.substring(2, dataIn.length());
  servo6Pos = dataInS.toInt();
  if (servo6PPos > servo6Pos) {
    for ( int j = servo6PPos; j >= servo6Pos; j--) {
      servo06.write(j);
      delay(30);
    }
```

```
      }
      if (servo6PPos < servo6Pos) {
        for ( int j = servo6PPos; j <= servo6Pos; j++) {
          servo06.write(j);
          delay(30);
        }
      }
      servo6PPos = servo6Pos;
    }
    // If button "SAVE" is pressed
    if (dataIn.startsWith("SAVE")) {
      servo01SP[index] = servo1PPos;  // save position into the array
      servo02SP[index] = servo2PPos;
      servo03SP[index] = servo3PPos;
      servo04SP[index] = servo4PPos;
      servo05SP[index] = servo5PPos;
      servo06SP[index] = servo6PPos;
      index++;                    // Increase the array index
    }
    // If button "RUN" is pressed
    if (dataIn.startsWith("RUN")) {
      runservo();  // Automatic mode - run the saved steps
    }
    // If button "RESET" is pressed
    if ( dataIn == "RESET") {
      memset(servo01SP, 0, sizeof(servo01SP)); // Clear the array data to 0
      memset(servo02SP, 0, sizeof(servo02SP));
      memset(servo03SP, 0, sizeof(servo03SP));
      memset(servo04SP, 0, sizeof(servo04SP));
      memset(servo05SP, 0, sizeof(servo05SP));
      memset(servo06SP, 0, sizeof(servo06SP));
      index = 0;  // Index to 0
    }
  }
}

// Automatic mode custom function - run the saved steps
void runservo() {
  while (dataIn != "RESET") {   // Run the steps over and over again until "RESET" button is pressed
```

```
for (int i = 0; i <= index - 2; i++) {  // Run through all steps(index)
 if (Bluetooth.available() > 0) {       // Check for incomding data
  dataIn = Bluetooth.readString();
  if ( dataIn == "PAUSE") {             // If button "PAUSE" is pressed
    while (dataIn != "RUN") {           // Wait until "RUN" is pressed again
      if (Bluetooth.available() > 0) {
        dataIn = Bluetooth.readString();
        if ( dataIn == "RESET") {
          break;
        }
      }
    }
  }
  // If speed slider is changed
  if (dataIn.startsWith("ss")) {
    String dataInS = dataIn.substring(2, dataIn.length());
    speedDelay = dataInS.toInt(); // Change servo speed (delay time)
  }
 }
 // Servo 1
 if (servo01SP[i] == servo01SP[i + 1]) {
 }
 if (servo01SP[i] > servo01SP[i + 1]) {
  for ( int j = servo01SP[i]; j >= servo01SP[i + 1]; j--) {
    servo01.write(j);
    delay(speedDelay);
  }
 }
 if (servo01SP[i] < servo01SP[i + 1]) {
  for ( int j = servo01SP[i]; j <= servo01SP[i + 1]; j++) {
    servo01.write(j);
    delay(speedDelay);
  }
 }

 // Servo 2
 if (servo02SP[i] == servo02SP[i + 1]) {
 }
 if (servo02SP[i] > servo02SP[i + 1]) {
```

```
  for ( int j = servo02SP[i]; j >= servo02SP[i + 1]; j--) {
    servo02.write(j);
    delay(speedDelay);
  }
}
if (servo02SP[i] < servo02SP[i + 1]) {
  for ( int j = servo02SP[i]; j <= servo02SP[i + 1]; j++) {
    servo02.write(j);
    delay(speedDelay);
  }
}

// Servo 3
if (servo03SP[i] == servo03SP[i + 1]) {
}
if (servo03SP[i] > servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j >= servo03SP[i + 1]; j--) {
    servo03.write(j);
    delay(speedDelay);
  }
}
if (servo03SP[i] < servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j <= servo03SP[i + 1]; j++) {
    servo03.write(j);
    delay(speedDelay);
  }
}

// Servo 4
if (servo04SP[i] == servo04SP[i + 1]) {
}
if (servo04SP[i] > servo04SP[i + 1]) {
  for ( int j = servo04SP[i]; j >= servo04SP[i + 1]; j--) {
    servo04.write(j);
    delay(speedDelay);
  }
}
if (servo04SP[i] < servo04SP[i + 1]) {
  for ( int j = servo04SP[i]; j <= servo04SP[i + 1]; j++) {
```

```
    servo04.write(j);
    delay(speedDelay);
  }
}

// Servo 5
if (servo05SP[i] == servo05SP[i + 1]) {
}
if (servo05SP[i] > servo05SP[i + 1]) {
  for ( int j = servo05SP[i]; j >= servo05SP[i + 1]; j--) {
    servo05.write(j);
    delay(speedDelay);
  }
}
if (servo05SP[i] < servo05SP[i + 1]) {
  for ( int j = servo05SP[i]; j <= servo05SP[i + 1]; j++) {
    servo05.write(j);
    delay(speedDelay);
  }
}

// Servo 6
if (servo06SP[i] == servo06SP[i + 1]) {
}
if (servo06SP[i] > servo06SP[i + 1]) {
  for ( int j = servo06SP[i]; j >= servo06SP[i + 1]; j--) {
    servo06.write(j);
    delay(speedDelay);
  }
}
if (servo06SP[i] < servo06SP[i + 1]) {
  for ( int j = servo06SP[i]; j <= servo06SP[i + 1]; j++) {
    servo06.write(j);
    delay(speedDelay);
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Ремарчук_Владислав Олегович

Тема: Автоматизована системи зарядки дронів

Спеціальність: 151 «Автоматизація та комп'ютерно-інтегровані технології»

Обсяг кваліфікаційної роботи:

Кількість сторінок записки 57

1. Короткий зміст роботи та прийнятих рішень: метою виконання роботи є розробка та впровадження системи зарядки акумуляторних дронів для зменшення часу заміни батареї та не активності дрону .....

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено аналіз відомих засобів та рішень. У другому розділі виконано проектування автоматизованої системи зарядки дронів. У програмно-апаратну реалізацію автоматизованої системи зарядки дронів .....

4. Позитивні сторони роботи: Використання існуючих функціональних елементів дронів. Використання простого позиціонування маніпулятору, та врахування можливих змін будови пристрою.

5. Негативні сторони роботи: у роботі недостатньо уваги приділяється розробці додаткового методу маніпуляції

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: відсутні

9. Оцінка дипломної роботи:

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) завідувач кафедри комп'ютерних наук, д.т.н., професор Бєрлан Олександр Володимирович

"15" червня 2025 р. (підпис)

Завідувачу кафедри АКІТтаР
д-ру техн.наук, проф. Мартинюку В.В.

Ремарчука Владислава Олеговича

ПІБ здобувача вищої освіти

ФІТ, 3 курс, групи АКІТс-22-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (StrikePlagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

02.06.25

дата

підпис

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Владислав РЕМАРЧУК

Співавтор:

Назва: Ремарчук Антиплагіат

Експерт:

Підрозділ: Кафедра автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

Коефіцієнт подібності 1:1%

Коефіцієнт подібності 2:0.3%

Мікропробіли: 0

Заміна букв: 15

Інтервали: 0

Білі знаки: 1

Дата створення звіту: 2025-06-15 18:26:21.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-15 Доцент Микола Федула

Дата експерт

Sun Jun 15 19:11:48 EEST 2025, Федула Микола Васильович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 1.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 13%**

	Document		Sum coincidence on the DB	
ID: 245954 Title: БКР Автоматизована система зарядки дронів Added in a DB: 2025-06-15 Authors: Владислав РЕМАРЧУК Heads: Ірина ФОРКУН Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	56129	847	758 (1%)	10 (1%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ АВТОМАТИЗАЦІЇ, КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:
Назва: Автоматизована системи зарядки дронів
Автор: Ремарчук Владислав Олегович
Спеціальність: 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма: Освітньо-професійна програма «Автоматизація та комп'ютерно-інтегровані технології»
Науковий керівник: Форкун Ірина Валеріївна, доктор технічних наук, професор
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:
Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:
1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій у переліку джерел посилання;
2) усі запозичення є фрагментарними або мають належним чином оформленні посилання;
3) виявлені модифікації тексту не впливають на відсоток схожості.
Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 1% і адресується до 19 джерел, що, з урахуванням наведених обгрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Завідувач кафедри _____ Валерій МАРТИНЮК
Гарант освітньої програми _____ Юрій ФОРКУН
Керівник кваліфікаційної роботи _____ Ірина ФОРКУН